TRABALHO DE GRADUAÇÃO

CONTROLE ÓTIMO PARA GRUPO DE ELEVADORES USANDO PROGRAMAÇÃO DINÂMICA

Renan Vargas Ávila

Brasília, Julho de 2019

ENGENHARIA MECATRÔNICA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

CONTROLE ÓTIMO PARA GRUPO DE ELEVADORES USANDO PROGRAMAÇÃO DINÂMICA

Renan Vargas Ávila

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Guilherme Caribé Carvalho, ENE/UnB _____
Orientador

Prof. Reinaldo Crispiniano Garcias, EPR/UnB _____
Co-orientador

Prof. Daniel Maurício Muñoz Arboleda, FGA/UnB _____
Examinador interno

Brasília, Julho de 2019

FICHA CATALOGRÁFICA

Renan Vargas Ávila
Controle ótimo para grupo de elevadores usando programação dinâmica

[Distrito Federal] 2019.

x, 101p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2019). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Sistema de Elevadores
2. Simulador *Arena*
3. Programação Dinâmica

I. Mecatrônica/FT/UnB
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

ÁVILA, RENAN VARGAS, (2019). Controle ótimo para grupo de elevadores usando programação dinâmica. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$015, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 101p.

CESSÃO DE DIREITOS

AUTOR: Renan Vargas Ávila

TÍTULO DO TRABALHO DE GRADUAÇÃO: Controle ótimo para grupo de elevadores usando programação dinâmica.

GRAU: Engenheiro ANO: 2019

Renan Vargas Ávila

SQN 113, bloco A, nº 601.

71000-000 Brasília – DF – Brasil.

Dedicatória

Dedico esse trabalho a Deus, o único onipotente, onisciente e onipresente e que ainda assim tem estado ao meu lado. Dedico também a minha família, amigos e irmãos da igreja que têm sempre me incentivado.

Renan Vargas Ávila

Agradecimentos

Agradeço primeiramente a Deus que por sua misericórdia e infinito amor me permitiu chegar até aqui. Agradeço a toda minha família que tem sempre me apoiado e orado por mim, em especial aos meus pais, Márcia e Robison, que sempre encorajaram a prosseguir, a minha namorada Gabriella que esteve me acompanhando de perto sempre atenciosa e me incentivando e também a Ramzi Fayad que me ajudou diretamente na parte prática do trabalho tirando dúvidas sobre a implementação.

Renan Vargas Ávila

RESUMO

O objetivo principal desse trabalho é a otimização do atendimento às chamadas feita por um grupo de elevadores que possuem DCS. A abordagem escolhida é a programação dinâmica que busca a solução ótima para este problema. O software Arena é utilizado como simulador da movimentação dos passageiros e dos carros os quais são organizados pelo otimizador em Visual Basic. Este sistema gerou os tempos de espera, viagem e destino que foram comparados com outros autores, obtendo-se uma média de 64,59 segundos para o tempo de espera, 75,2 segundos para vôo e 139,8 segundos de destino.

ABSTRACT

The main obective of this work is the call otimization attendance made by a group of elevators that has DCS. The chosen approach is dynamic programmig which looks for the optimal solution to the problem. The Arena software is used as the simulator for the movement of the people and the cars, which are organized by the optimizer in Visual Basic. This system generated the wait, jouney and destiny times which were compared with anothers authors, getting an avarage of 64,59 secods for waiting time, 75,2 seconds for journey and 139,8 to reach the destination.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

AWT	Tempo Médio de Espera - *Average Waiting Time*
INT	Intervalo Médio - *Average Interval*
CPU	Unidade Central de Processamento - *Central Processing Unit*
DCS	Sistema de Controle de Distino - *Destiny System Control*
OLE	*Object Linking and Embedding*
WT	Tempo de espera - *Waiting Time*
JT	Tempo de viagem - *Journey Time*
VBA	*Visual Basic for Applications*

Capítulo 1

Introdução

1.1 Contextualização

A verticalização das construções tem levado a uma preocupação crescente com o deslocamento entre os andares, o qual pode ser feito por meio de escadas normais ou rolantes ou elevadores. Este tipo de avaliação depende da estrutura interna do prédio e também da capacidade de cada meio de transporte.

Entretanto quando se trata de prédios altos, o uso de elevadores torna-se quase imprescindível dado que não é necessário esforço da pessoa como as escadas requerem e o seu deslocamento vertical permite percorrer diversos andares diretamente, o que não acontece nas escadas rolantes que possuem uma trajetória em diagonal.

O avanço da tecnologia permite a presença não de apenas um elevador, mas de um ou mais grupos deles os quais tornam o transporte ainda mais rápido.

1.2 Definição do problema

A utilização de grupo de elevadores requer um grande investimento desde o planejamento e construção do prédio até a parte elétrica e mecânica, portanto é necessário que este grupo seja capaz de oferecer um bom serviço. Todavia esta definição pode conter diversos parâmetros, como o tempo, o gasto de energia, conforto dos passageiros e investimento em infraestrutura.

Do ponto de vista dos passageiros, um sistema de elevadores satisfatório é aquele que atende às chamadas e transporta rapidamente. Isso pode ser alcançado por meio de melhorias nas partes mecânicas do elevador e um bom gerenciamento das chamadas. O primeiro item dependerá mais dos componentes físicos do elevador e consequentemente o quanto se está disposto em investir, esses tipos de análises não estarão presentes neste trabalho.

Já o segundo pode ser alcançado mediante métodos de otimização, para tanto é necessário conhecimento prévio do destino de cada pessoa antes que ela entre no elevador pois será papel

do DCS (*Destiny System Control*) decidir qual irá atender a chamada. Isto já representa uma limitação de diversos sistemas atuais pois operam com botões direcionais em cada andar como mostrado na Figura 1.1 a direita, e botões para cada um dos andares, Figura 1.1 a esquerda. Essa configuração faz com que a chamada seja atendida apenas pelo elevador evocado, a não ser que o passageiro chame mais de um elevador e embarque no primeiro que chegar, o que causará ainda mais atrasos para outros andares porque uma única chamada será atendida por mais de um elevador desnecessariamente.

Figura 1.1: Botões de elevadores tradicionais

O gasto de energia também pode ser minimizado através de um sistema de controle eficaz pois não ocorrerão casos onde o elevador chega em um andar onde a chamada já foi atendida, como descrito anteriormente. Outro ponto interessante é que um dos fatores a ser considerado pelo controle é o número de paradas, que ao ser reduzido, diminui a energia gasta com aceleração e desaceleração do elevador.

A configuração ideal para as chamadas é mostrada na Figura 1.2 pois dessa maneira o usuário inicialmente informa em um painel o seu destino e então ele é designado a um dos elevadores, sendo assim, não existe painel dentro da cabine, apenas botões para emergência ou comunicação.

Figura 1.2: Elevador com DCS implementado

1.3 Objetivos do projeto

1.3.1 Objetivo Geral

O projeto busca primeiramente criar um ambiente que simule a movimentação de pessoas em um grupo de elevadores durante o Uppeak. E então reduzir os tempos de espera e viagem dos usuários selecionando o melhor carro para cada passageiro por meio da programação dinâmica a partir de dados atuais e anteriores do sistema.

1.3.2 Objetivos Específicos

- **Otimizador**: Criar um algoritmo que, quando uma nova chamada é registrada, leia os dados atuais e anteriores da simulação, encontre o melhor elevador para atender a nova solicitação com a programação dinâmica e então armazene esta nova chamada na memória do elevador selecionado.
- **Gerador de Chamadas**: Criar um simulador de chamadas que primeiramente possui uma porção responsável por simular a chegada dos usuários no sistema, criando chamadas seguindo um padrão Uppeak as quais são gerenciadas pelo otimizador.
- **Movimentação dos Elevadores**: Criar um conjunto de rotas pelas quais o carro se movimenta de acordo com as chamadas alocadas em sua memória e alterando as variáveis do sistema (andar atual, direção...) afim de mantê-lo constantemente atualizado.
- **Movimentação dos Passageiros**: Após a criação dos usuários com seus respectivos atributos, é preciso garantir que eles esperem o elevador chegar na origem e depois também aguardar até alcançar o destino. Após tais simulações, são adquiridos os tempos de cada passageiros, informação essencial para a avaliação da qualidade do sistema.

1.4 Resultados obtidos

Com a utilização do arena obteve-se um tempo médio para um passageiro chegar ao seu destino de 136,68 segundos no caso onde o tráfego é o mais intenso possível, ou seja, com 16% da população do prédio no sistema considerando um local com 10 andares, 460 pessoas distribuídas uniformemente em cada andar e três elevadores. Este valor é composto de 69,27 segundos do tempo de espera até poder entrar no elevador e 67,41 segundos para o carro chegar no destino.

1.5 Apresentação do manuscrito

Este documento possui quatro capítulos, incluindo esta introdução, onde ocorre um primeiro contato com o assunto em pauta, apresentando o método considerado de fazer chamadas e a importância do estudo da otimização dos parâmetros do sistema. No segundo capítulo, fundamentos,

mostra tipos diferentes de carros os quais servem para propósitos distintos devido a suas características peculiares e por consequência teriam uma modelagem diferente para serem analisados. Em sequência, no capítulo de fundamentos teóricos são citadas as funções e regras que um grupo de elevadores e passageiros devem seguir, os quais foram tidos como premissas parar o projeto do otimizador. Outras abordagens para este problema são mencionados brevemente, bem como uma avaliação da qualidade de um serviço. São apresentados alguns exemplos de situações, os cálculos dos seus tempos e também a conceituação da programação dinâmica em encontrar o ponto ótimo e os softwares utilizados neste trabalho.

Metodologia inicia com a adaptação do método para este problema, bem como com a modelagem do sistema no *Arena* tanto do gerador de chamadas quanto da dinâmica dos elevadores. Então é descrito, de forma resumida a sequência de ações do otimizador e onde são armazenados os dados de saída que serão utilizados no restante da simulação.

Ao final são apresentados os resultados e a forma como foram adquiridos, bem como seus respectivos gráficos e comparações com trabalhos na mesma área afim de avaliar a eficiência da abordagem utilizada.

Capítulo 2

Fundamentos

2.1 HISTÓRICO DO DESENVOLVIMENTO DOS ELEVADORES

Desde os primórdios o ser humano buscava uma maneira eficiente para transportar cargas e passageiros verticalmente (History of elevators, 2018) o que levou ao desenvolvimento dos primeiros elevadores que eram chamados de gruas. Movidos a tração humana, animal ou as vezes hidráulica, esses mecanismos eram usados desde o início do III a.C.

A partir de 1800 foram desenvolvidos elevadores que utilizavam a pressão da água que era usado para elevar materiais em fábricas, minas e armazéns. Então em 1852 Elisha Graves Otis introduziu dispositivos de segurança no elevador o que; aliado ao desenvolvimento das fontes de energia; popularizou o transporte de pessoas por este meio. O primeiro elevador elétrico foi construído por Wener Von Siemens em 1880, nove anos depois o primeiro elevador deste tipo fora instalado comercialmente.

2.2 MODELOS ESPECIAIS

Com o aumento do número de andares dos prédios e consequentemente a quantidade de pessoas presentes nele, foi necessário o surgimento de novas opções de elevadores afim de melhor atender as diferentes configurações dos edifícios.

Elevadores com plataforma dupla comportam dois carros de passageiros conectados como apresentado na Figura 2.1, onde uma das plataformas atende os andares ímpares e outra os pares. A redução do número de carros, velocidade nominal, número de paradas e tamanho dos carros são algumas das vantagens deste modelo, todavia existem também desvantagens como a utilização errada dos passageiros, necessidade da demanda ser balanceada entre andares ímpares e pares, distância entre andares deve ser regular entre outras.

Em hotéis, shoppings e escritórios é comum a presença de elevadores de observação, Figura 2.2 porque eles oferecem uma vista panorâmica do local, por esta razão, geralmente possuem uma velocidade nominal inferior fazendo com que a duração da viagem seja maior. As portas podem

Figura 2.1: Elevador de plataforma dupla

ser maiores para facilitar a passagem de crianças e idosos e geralmente possuem um tempo de abertura e fechamento maior para garantir a segurança por serem de vidro. Esses aspectos levam a uma performance de 50% deste modelo em relação aos elevadores convencionais (Barney, 2003).

Figura 2.2: Elevador de observação

2.3 TÉCNICAS DE CONTROLE PARA GRUPOS DE ELEVADORES

Inicialmente é essencial o conhecimento das quatro tarefas primárias e cinco regras que um sistema de elevador deve cumprir (Barney, 2003), pois são a partir delas que ocorre o desenvolvimento dos métodos de otimização.

Tarefas Primárias:

- Prover igual serviço para todo andar do prédio.
- Minimizar o tempo gasto pelos passageiros esperando o serviço no embarque.
- Minimizar o tempo que os passageiros levam para mover de um andar para outro.
- Servir a quantidade de passageiros que for possível dentro de um determinado tempo.

É possível que aconteça conflito entre duas tarefas, por exemplo, prestar um serviço igual a todos os andares e minimizar o tempo de espera. Caso o edifício tenha uma demanda diferente em cada andar, essas duas tarefas se tornarão divergentes.

Regras:

- Um carro não pode reverter a direção de viagem com passageiros dentro.
- O elevador deve parar no andar de destino do passageiro.
- Os passageiros devem entrar no carro ao qual foram designados.
- Um elevador não deve parar em um andar onde não há passageiros desejando entrar ou sair.

2.3.1 ABORDAGENS TRADICIONAIS

Tradicionalmente são utilizadas técnicas mais simples, a mais comum é que utiliza os botões direcionais em cada andar onde o usuário pode indicar o sentido de sua viagem e dentro do carro um painel com todos os andares, portanto o sistema só saberá onde o passageiro irá, quando este já estiver dentro do elevador. Este tipo de abordagem trata cada carro individualmente, ou seja, independente de onde os outros estão, se seu botão for pressionado, ele irá para o andar onde foi requisitado. Isto pode gerar muitos desperdícios de tempo tendo em vista que um usuário, desejando ser atendido o mais rápido possível, pode vir a chamar todos os elevadores de uma vez só, fazendo o sistema como um todo se direcionar para uma solicitação que será atendida por apenas um carro, os demais que chegarem no andar, apenas perderão tempo.

Devido às limitações impostas pelos controles tradicionais, foram desenvolvidas novas maneiras de gerenciar as chamadas, não mais tratando cada elevador individualmente, mas sim o grupo deles, analisando o sistema como um todo para então tomar uma decisão que melhor sirva aos passageiros,

ou seja, que reduza os tempos de espera pelo carro e o tempo de viajem. Para avaliar o sistema, algumas técnicas de otimização se valem de funções custo que são essenciais para as tomadas de decisão, os parâmetros considerados nesta função, são os mesmos do grupo de elevadores, carga do elevador, origem da chamada, destino, direção... A ideia central é maximizar ou minimizar esta função custo afim de encontrar a melhor solução para o problema dos elevadores. A seguir são citadas algumas maneiras de controlar este sistema considerando um grupo de elevadores.

2.3.2 ENXAME DE PARTÍCULAS

Desenvolvido pelo psicólogo James Kennedy e pelo engenheiro Russell Eberhart (1995), este algoritmo se baseia na busca de alimento por bandos de pássaros e também cardumes de peixes. Inicialmente cada indivíduo procura por comida aleatoriamente, mas ao comunicar-se entre si, o grupo segue o indivíduo que esteja mais próximo da melhor fonte de comida. O enxame de partículas não utiliza da seleção, portanto normalmente todos os indivíduos sobrevivem do início da procura até o final. No algoritmo, esta decisão é feita avaliando cada solução por meio de uma função custo que relaciona os parâmetros a serem otimizados como o tempo de viagem, número de paradas, distância...

Atualmente existem variações baseadas no comportamento em grupo de outros seres como é o caso do ABC (*Artificial Bee Colony*) onde as abelhas compartilham informações sobre a fonte de alimento encontrada através de uma dança. Outro grupo são os vaga-lumes que tem seu comportamento descrito através da luminescência de seus corpos, onde a partir da intensidade da luz, é possível criar a função custo.

2.3.3 LÓGICA NEBULOSA

A teoria dos conjuntos nebulosos propõe graus de pertinência de um elemento a um determinado conjunto (1965), que varia de 0 a 1. Em um sistema baseado em lógica nebulosa, essas atribuições são dadas pelas funções de pertinências e nelas são baseados os blocos de fuzificação os quais recebem os dados reais. Após o tratamento dos dados, eles precisam retornar ao domínio do mundo real que é responsabilidade do bloco de defuzificação, Figura 2.3 mostra o diagrama do sistema de forma resumida.

A base de conhecimento é o modelo usado para representar o sistema real e a inferência é onde ocorre as comparações dos acontecimentos e as tomadas de decisões. Estas escolhas decidem o elevador mais conveniente para atender as chamadas baseadas em um valor de prioridade.

2.3.4 ALGORITMO GENÉTICO

Apesar do grande avanço da capacidade de processamento atual, nem sempre é possível calcular todas as casos possíveis, no caso dos elevadores, todas as rotas. No entanto ainda é possível encontrar uma solução ótima ótimo por meio de algoritmos genéticos (Siikonen, 2000) que seguem a lógica da seleção natural, a qual seleciona os melhores genes vindos dos pais da nova geração.

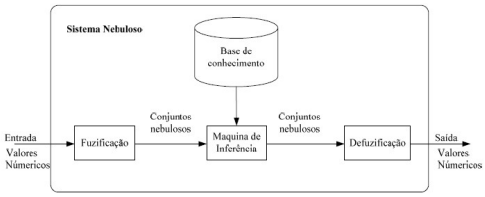


Figura 2.3: Sistema Nebuloso (Alvares, 2010)

Em um caso onde existem quatro elevadores e quatro chamadas para embarque, como mostrado na Figura 2.4. Cada carro alocado a uma determinada chamada é um gene e o conjunto de todos os quatro com suas quatro respectivas chamadas formam um cromossomo. Inicialmente as rotas são geradas aleatoriamente e então as melhores são escolhidas, por meio da herança, cruzamento e mutação é gerada a nova geração. Após algumas iterações, a solução converge para a solução ótima.

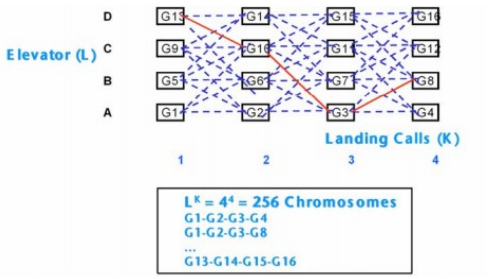


Figura 2.4: Alocações possíveis entre cada um dos elevadores e cada uma das chamadas (Siikonen, 2000)

2.3.5 PROGRAMAÇÃO DINÂMICA

Este método constitui em reduzir o problema em um conjunto de estados em cada iteração e um custo para a transição entre estados. A modelagem do sistema pode ser feitas de diversas maneiras, Luran Li (2014), por exemplo, considera um estado sendo;

$$S_t = (R_{ta}, D_{tb}) \tag{2.1}$$

Onde:

t = 0,1,2...T;

a = Próximo andar, tempo de chegada, carga k para o andar j;

b = (i,j) Chamada do andar i para o j;

R_{ta} = Número de elevadores com atributo a no tempo t;

D_{tb} = Número de passageiros com o atributo b no tempo t.

Quanto a decisão a ser tomada em cada iteração, é considerado:

$$x_t = (x_{tad}) \tag{2.2}$$

Onde:

t = 0,1,2...T;

a = Atributo do elevador;

d = (i,j) ação de responder à chamada do andar i ao j.

Por fim, tem-se o objetivo:

$$V_t(S_t) = min(C(S_t, x_t) + \gamma E(V_{t+1}(S_{t+1})|S_t)) \tag{2.3}$$

Sendo:

$C(S_{yt}, x_t)$ = Custo de tomar a decisão x_t no estado S_t;

$E(V_{t+1}(S_{t+1})|S_t)$ = Esperança de tomar a decisão x_t em S_t;

γ = Fator de desconto ($0 < \gamma < 1$).

Através desta abordagem, a autora verificou que o tempo para requerido para resolver este problema cresce exponencialmente devido aos diversos estados, custos e incertezas possíveis. Para contornar isto foi considerado uma parcela do problema e melhorado a solução de maneira iterativa conseguindo assim, uma porcentagem de otimização semelhante em diferentes demandas.

Popyne e Cassandras (1997) usaram programação dinâmica para encontrar o limite da política ótima em uma condição matutina de Uppeak. De acordo com a sua política, o elevador é despachado quando o limite de passageiros é atingido. Mais recentemente Wesselowski e Cassandras (2006) partiram para uma abordagem que avalia se vale a pena ou não o elevador parar em um andar particular ou seguir a trajetória, isto requer uma precisa capacidade de decisão em tempo real.

Foi escolhida esta abordagem para o problema dos elevadores pois algoritmos bioinspirados e lógica nebulosa já haviam sido aplicadas na Universidade de Brasília por alunos de mestrado. Tendo em mãos condições iniciais próximos, foi trabalhado em cima da programação dinâmica já que não tinha sido explorada nesta universidade.

2.4 MODELAGEM

Antes de encontrar a função custo do problema de programação dinâmica é preciso definir alguns conceitos relacionados ao tráfego de elevadores. Também deve-se garantir que as características do sistema sejam mantidas o mais próximo possível dos trabalhos anteriores para que a comparação de resultados seja consistente. []

2.4.1 DEMANDA PELO SERVIÇO

Um mesmo prédio pode possuir diferentes populações em cada andar, isso leva a uma demanda do serviço de forma desigual. Além disso podem existir regiões de acesso restrito, tornando o tráfego de passageiros ainda mais desbalanceado.

A intensidade no uso dos elevadores também muda durante o dia, essencialmente existem três períodos com padrões predominantes: Uppeak, Interfloor e Downpeak. Essas demandas estão representadas na Figura 2.5.

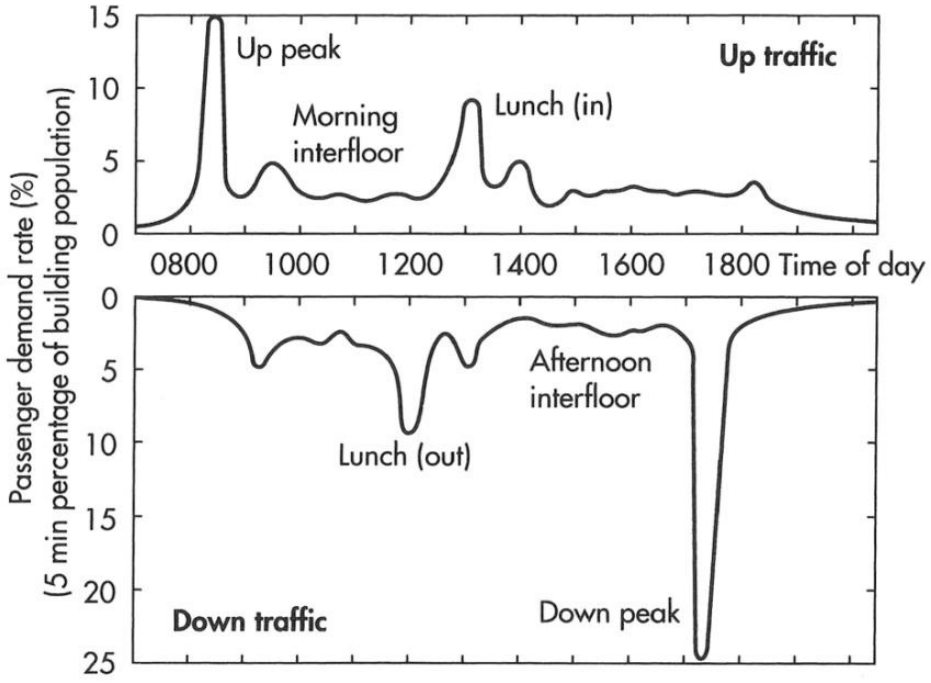


Figura 2.5: Demanda do serviço (Barney, 2003)

Uppeak

Ocorre quando a maior parte ou 100% do fluxo de passageiros é para cima, onde a maioria dessas pessoas vem do andar principal que é geralmente o térreo, o acesso ao prédio. Portanto esse comportamento é frequente durante a manhã pois é o momento em que os empregados se dirigem aos seus respectivos locais de trabalho.

A Figura 2.6 mostra a taxa de chegada durante uma hora onde é possível ver um aumento gradual do tráfego até o pico que dura cinco minutos e depois cai rapidamente. O sistema de elevadores deve estar preparado para suportar esse curto período de pico sem causar muitos transtornos às pessoas.

Downpeak

Acontece geralmente no final do dia, quando os funcionários estão saindo do edifício pois é caracterizado pelo grande fluxo dos andares para o térreo. Neste caso, a capacidade dos sistema aumenta 50% porque um carro fica completamente cheio ao passar por três, quatro ou cinco

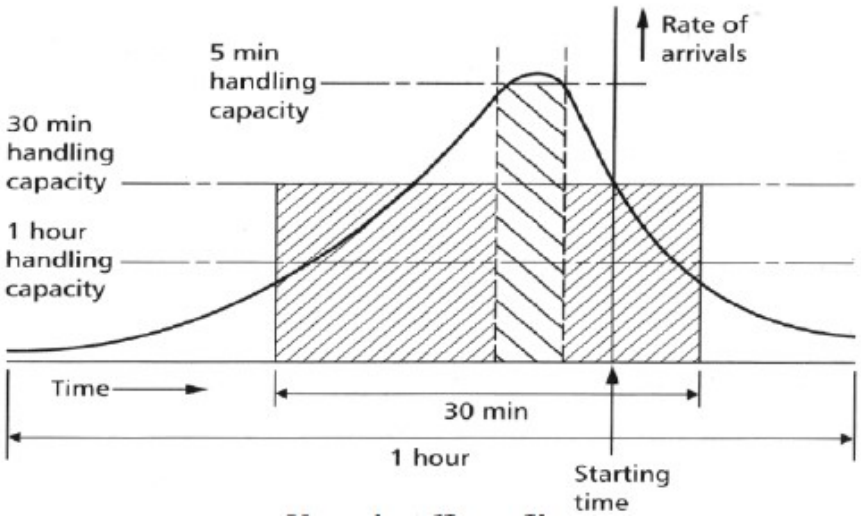


Figura 2.6: Taxa de utilização durante uma hora de Uppeak (Barney, 2003)

andares, depois ele faz uma viagem expressa para o terminal principal, isso reduz o número de paradas durante a viagem.

De maneira semelhante ao uppeak, a Figura 2.7 apresenta em mais detalhes o perfil do down-peak durante uma hora, a diferença é que neste caso o maior uso ocorre durante dez minutos.

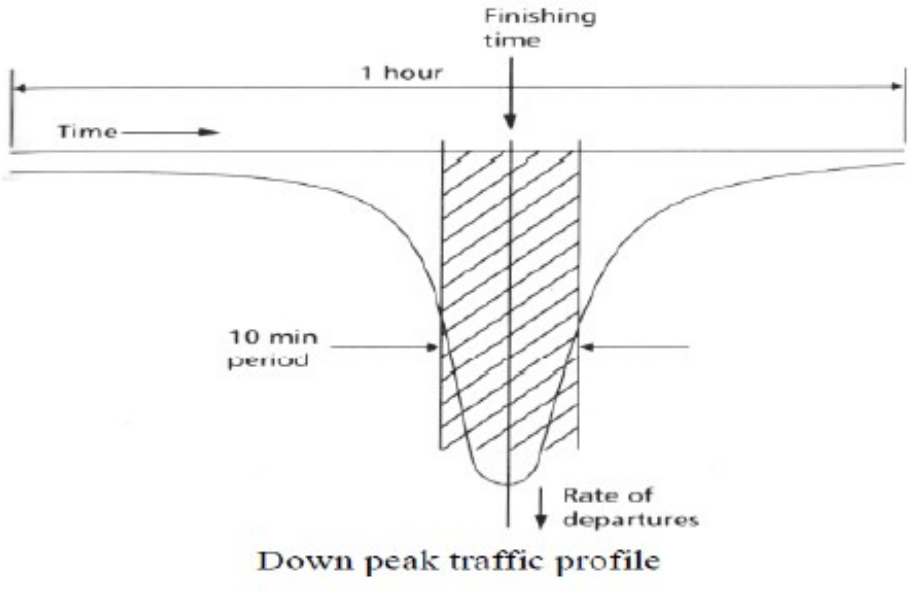


Figura 2.7: Taxa de utilização durante uma hora de Downpeak (Barney, 2003)

Interfloor

Este tipo de movimentação não segue qualquer tipo de padrão e ocorre em qualquer tipo de prédio. Durante o interfloor a demanda é menor em relação as anteriores.

2.4.2 QUALIDADE DO SERVIÇO

Para o usuário, a qualidade do serviço é mensurado pelo tempo de espera tanto de ser atendido quando de chegar ao destino. O primeiro é contabilizado desde o momento em que a chamada é realizada até a entrada do indivíduo no carro, este parâmetro é chamado de WT (*Waiting Time*). A Tabela 2.1 mostra o porcentagem da população total que chega e o tempo de espera bom para cada tipo de prédio, esses dados vêm dos cinco minutos de uppeak.

Tabela 2.1: Porcentagem da taxa de chegada (Barney, 2003)

Tipo de Prédio	**Taxa de Chegada**	**Tempo de Espera (s)**
Hotel	10-15%	30-50
Apartamento	5-7%	40-90
Hospital	8-10%	30-50
Escola	15-25%	30-50
Escritório Locação Multipla		
Regular	11-15%	25-30
Prestígio	17%	20-25
Escritório Locação Única		
Regular	15%	25-30
Prestígio	17-25%	20-25

Todavia não apenas a taxa de chegada de pessoas influência no tempo de espera, mas a quantidade de passageiros presentes dentro do elevador, definido por RC (*Rated Car Capacity*), influencia na relação WT/INT, onde INT (*Average Interval*) é intervalo médio de tempo entre a chegada de um elevador e outro. É convencionado que para um carro RC = 80% e AWT = 0,85INT, entretanto para $RC \geqslant 90\%$, o tempo de espera se estende até 130% do intervalo calculado e com RC = 100%, AWT tende a infinito, estes valores estão presente na Tabela 2.2.

Tabela 2.2: Performance no Uppeak (Barney, 2003)

Carga do Elevador (%)	**AWT/INT**	**Carga do Elevador (%)**	**AWT/INT**
30	0,32	75	0,74
40	0,35	80	0,85
50	0,40	85	1,01
60	0,50	90	1,30
70	0,65	95	1,65

2.4.3 CARACTERÍSTICAS DO PRÉDIO ESTUDADO

Para fins de comparação com outros trabalhos na área, será considerado um prédio comercial que possui o sistema DCS implementado. Para facilitar o cálculo dos parâmetros a serem otimizados será avaliado um prédio tem uma mesma demanda para todos os andares com a maioria das chamadas partindo do térreo pois se considera um regime Uppeak e o elevador possuirá uma carga

máxima de oito pessoas. Tais características foram as mesmas utilizadas pelo Rodriguez (2015) para uma comparação mais fidedigna.

Tabela 2.3: Dados do edifício estudado (Alvaro, 2010)

Tipo de prédio	Comercial
População	460 pessoas
Número de andares	10
Número de elevadores	3

2.4.4 FUNÇÃO CUSTO DO ELEVADOR

A função custo é uma equação que avalia a alocação de recursos por meio dos custos de cada decisão a ser tomada durante o processo. Este conceito é vastamente utilizado em *Machine Learning* buscando resultados melhores em cada iteração do algoritmo, também é aplicada em diversas áreas da pesquisa operacional, alguns exemplos seriam realocação de trabalhadores dentro em diversas fábricas com características particulares, soluções ótimas para problemas relacionados a tráfego, dentre outros.

Sendo o tempo de espera do passageiro um dos parâmetros a ser otimizado, é necessário entender as possibilidades e considerar todos os tempos envolvidos no translado os quais dependem da movimentação do carro, do sentido da chamada e das chamadas alocadas previamente. Antes de fazer esta análise é preciso saber que T_a é o tempo que o elevador leva para se deslocar entre os andares, T_{ac} aceleração ou desaceleração, T_p abertura ou fechamento de portas e T_{es} a entrada e saída de passageiros do carro. Ainda é possível considerar outras variáveis a serem otimizadas tais como eficiência energética, quantidade de paradas, utilização do carro, dentre outros.

Quando um elevador está em movimento para atender uma chamada, existem inicialmente quatro possibilidades que podem ocorrer, (para estes primeiros exemplos, está sendo considerado que o carro está na origem do elevador atendendo a chamada):

- **Caso 1**: Quando tanto o elevador e a chamada têm o mesmo sentido estando a trajetória do segundo contido no primeiro.
- **Caso 2**:Quando tanto o elevador e a chamada têm o mesmo sentido, mas o destino da chamada está fora do trajeto do carro.
- **Caso 3**:Quando tanto o elevador e a chamada têm o mesmo sentido, mas a origem da chamada está fora do trajeto do carro.
- **Caso 4**: Quando o elevador e a chamada têm sentidos opostos.

Custo temporal do caso 1 Ao ser realizada a chamada, é preciso calcular o tempo do elevador se deslocar do andar 1 (Origem do elevador) até o andar 3 (Origem da chamada), este será o WT. A próxima parte do custo é calculado como o tempo do trajeto do andar 3 até o 7

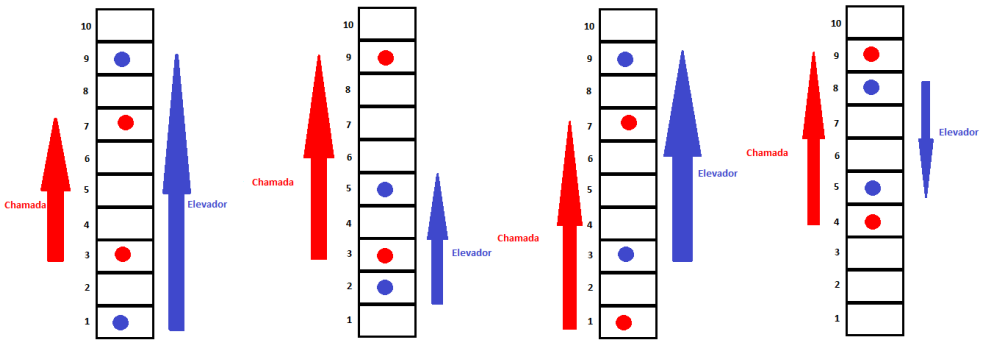


Figura 2.8: Quatro casos de chamadas com elevador em movimento, do caso 1 ao 4 da esquerda para a direita respectivamente

(rota da chamada) que equivale ao JT, este também pode ser encontrado subtraindo o custo para chegar na origem ($Custo_3$) do custo para chegar no destino da nova chamada ($Custo_7$). Este caso pode ser visualizado na primeira situação da Figura 2.8.

$$WT = (3-1)T_a + 2T_{ac} + 2T_p \tag{2.4}$$

$$JT = (7-3)T_a + 2T_{ac} + 2T_p = Custo_7 - Custo_3 \tag{2.5}$$

Custo temporal do caso 2 Este cenário é semelhante ao anterior, sendo necessário calcular WT que é o tempo para chegar na origem da chamada partindo do segundo andar. Já o JT será a soma do custo para chegar até o andar 5, o tempo da parada e o tempo para alcançar o destino da chamada partindo do quinto andar ($Custo_5$).

$$WT = (3-2)T_a + 2T_{ac} + 2T_p \tag{2.6}$$

$$JT = ((9-5)T_a + 2T_{ac} + 2T_p + T_{es}) + Custo_5 \tag{2.7}$$

Custo temporal do caso 3 Quando a origem da chamada está fora do trajeto original do elevador é gerado um problema devido a segunda regra do controle de elevadores no tópico 2.3 pois o elevador não pode mudar de direção com pessoas dentro, sendo assim o cálculo de custo será diferente. Primeiramente se calcula o tempo da rota do elevador, andar 3 ao 9 ($Custo_9$), depois a penalidade do retorno até a origem da chamada (andar 9 ao 1) e por fim calcular o tempo gasto para atender esta chamada (andar 1 ao 7). De maneira mais resumida, o WT será o custo para chegar até o nono andar somado ao tempo de descida até o térreo, já a viagem será simplesmente o tempo que levará para ir diretamente do andar 1 ao 7 já que não há paradas intermediárias.

$$WT = Custo_9 + T_{es} + (9-1)T_a + 2T_{ac} + 2T_p \tag{2.8}$$

$$JT = (7-1)T_a + 2T_{ac} + 2T_p \tag{2.9}$$

Custo temporal do caso 4 Neste caso só será possível alocar a chamada depois que o elevador completar o trajeto original dele, entretanto diferente do caso anterior, sua penalidade será menor, pois precisará voltar apenas do andar 9 ($Custo_9$) ao andar 8 (origem da nova chamada);

$$WT = Custo_9 + T_{es} + (9-8)T_a + 2T_{ac} + 2T_p \tag{2.10}$$

$$JT = (8-5)T_a + 2T_{ac} + 2T_p \tag{2.11}$$

Um ponto importante a ser ressaltado é que todos os exemplos dados anteriormente consideram apenas duas trajetórias, a do elevador e da nova chamada. Caso existam mais paradas já alocadas na memória que serão tratadas antes da nova chamada, é necessário somá-las ao custo para chegar na origem da nova chamada. Vejamos o caso apresentado na Figura 2.9, o elevador está no andar 5, que é origem de uma das suas chamadas da memória, e já está com as portas abertas depois dela atenderá a outra chamada do andar 8 ao 10. Neste exemplo, o tempo de espera para atender essa nova chamada 1-3 será o período que levará para se fazer o percurso de 5-6-8-10-1 pois não foi possível alocar a nova solicitação entre as já existentes, como ocorre no 1 da Figura 2.8, ou seja, a nova solicitação foi colocada como última na fila de pendências. Portanto, o tempo para o elevador chegar até a origem O_1 é dada pode ser decomposto em quatro intervalos, que somados resultam no tempo de espera para o atendimento da chamada.

$$WT_6 = (6-5)T_a + 2T_{ac} + 2T_p + T_{es} \tag{2.12}$$

$$WT_8 = WT_6 + (8-6)T_a + 2T_{ac} + T_p + T_{es} \tag{2.13}$$

$$WT_{10} = WT_8 + (10-8)T_a + 2T_{ac} + 2T_p + T_{es} \tag{2.14}$$

$$WT_1 = WT_{10} + (10-1)T_a + 2T_{ac} + 2T_p \tag{2.15}$$

$$JT_{1-3} = (1-3)T_a + 2T_{ac} + 2T_p \tag{2.16}$$

Ainda considerando chamadas previamente alocadas na memória do elevador, há ainda um último caso, semelhante ao anterior, entretanto neste é possível alocar a nova chamada entre as já existentes. Isso acontece quando o carro vai tanto passar pela origem e pelo destino quanto o fará no mesmo sentido. Portanto o elevador já faria aquele trajeto inevitavelmente, todavia agora ele terá de parar nestes andares.

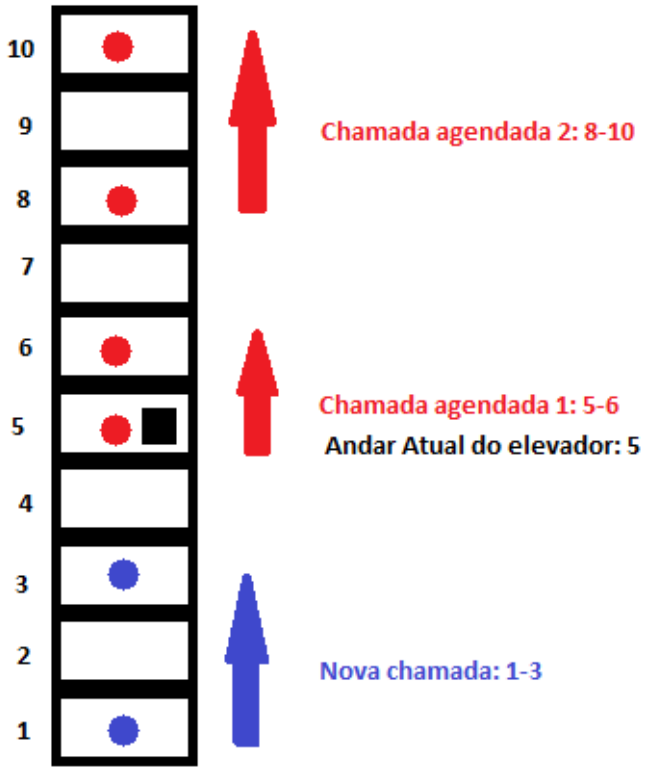


Figura 2.9: Caso onde a nova chamada foi alocada depois das paradas já existentes

Na Figura 2.10 o elevador está no andar 4, subindo para atender ao pedido de 5-10, depois atender a solicitação do 1-3. Inevitavelmente o elevador irá percorrer o trajeto 8-6, portanto é possível alocá-la entre estas duas, não precisando esperar o elevador atender ambas chamadas alocadas, isso faz o custo diminuir em relação à ideia de ir apenas empilhando as novas solicitações. Portanto o custo para este elevador responder a esse usuário pode ser novamente divido em três equações que serão somadas.

$$WT_5 = (5-4)T_a + T_{ac} + T_p \tag{2.17}$$

$$WT_{10} = WT_5 + (10-5)T_a + 2T_{ac} + 2T_p + T_{es} \tag{2.18}$$

$$WT_8 = WT_{10} + (10-8)T_a + 2T_{ac} + 2T_p + T_{es} \tag{2.19}$$

$$JT_{8-6} = (8-6)T_a + 2T_{ac} + 2T_p + T_{es} \tag{2.20}$$

Como dito anteriormente, os casos acima são válidos para quando o elevador está em movimento. O cálculo do tempo para quando o elevador estiver parado, sem chamadas pendentes,

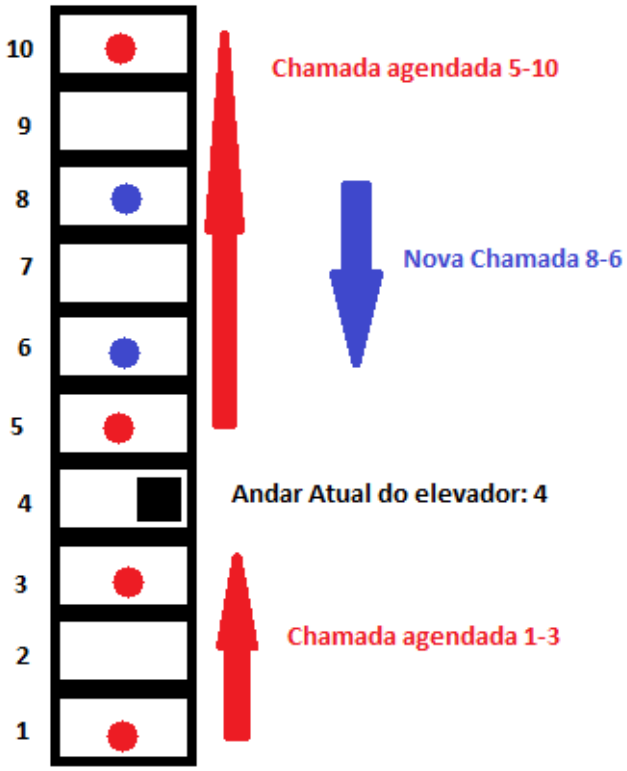


Figura 2.10: Caso onde a nova chamada foi alocada entre as paradas já existentes

dá-se simplesmente pelos tempos de aceleração, translado do andar atual até a origem e abertura de portas, sem quaisquer penalizações, já que não há trajetórias gravadas na memória anteriormente.

Os tempos utilizados acima, desaceleração (Tac), o deslocamento entre andares (Ta), abertura e fechamento de portas (Tp) e entrada e saída de passageiros (Tes) são valores característicos do próprio sistema de elevadores, exceto o tempo de entrada e saídas de pessoas, que depende da quantidade de gente querendo entrar e/ou sair. Para uma comparação mais fidedigna, os tempos utilizados nesta análise são os mesmos empregados no trabalho do Rodriguez (2015). Além do custo que o elevador tem até chegar à origem da chamada, há o tempo que o mesmo levará com a viagem JT_i. Esta informação está diretamente ligada à quantidade de paradas que o elevador terá de fazer ao longo do percurso. Idealmente o carro irá direto da origem ao destino, pois é o menor tempo possível, entretanto caso este elevador esteja atendendo ou irá atender a muitas outras pessoas, o tempo de viagem tende a aumentar. Este parâmetro é muito importante na função custo pois ele ajuda a equilibrar o sistema, tendo em vista que outros parâmetros não estão sendo considerados explicitamente na função custo como por exemplo a quantidade de pessoas em um elevador ou de paradas alocadas, todavia são contabilizados indiretamente em JT_i pois o aumento desses valores faz crescer o tempo de voo. No caso da quantidade de passageiros, este valor é utilizado para evitar que um elevador exceda a sua capacidade, adicionando uma penalidade grande o suficiente que independente do quão rápido ele possa atender a chamada, continua sendo

o pior a ser escolhido. Já a quantidade de paradas alocadas é usada como critério de desempate entre elevadores com o mesmo custo para atender a chamada.

Considera-se o exemplo ilustrado na Figura 2.11 com os três elevadores com chamadas previamente agendadas onde eles já atenderam à primeira parada no térreo e estão levando os passageiros para seus destinos, lembrando que está sendo considerado um padrão *Uppeak*, ou seja, a maioria das chamadas irá partir do térreo. O elevador 1 terá que parar no andar quatro e nove, o elevador 3 parará no andares quatro e sete, já o segundo elevador estava subindo para deixar o passageiro no terceiro andar quando foi alocada nele outra chamada, do térreo ao andar cinco, todavia como já estava a caminho do terceiro andar ele terá que deixar o usuário no destino e então retornar ao térreo. Nessa situação, a maioria das próximas chamadas será alocada no segundo carro pois inevitavelmente ele terá que voltar ao térreo e seu custo de retorno é o menor, já que os demais tem destinos mais distantes para atender, gerando uma penalidade de retorno maior. Portanto as chamadas rapidamente se acumularão no segundo carro, todavia isso vai aumentar o tempo de viagem para cada percurso tendo em vista o aumento das paradas, após um certo número de solicitações agendadas no segundo elevador, o custo será tão alto que os demais elevadores passarão a ser a melhor opção.

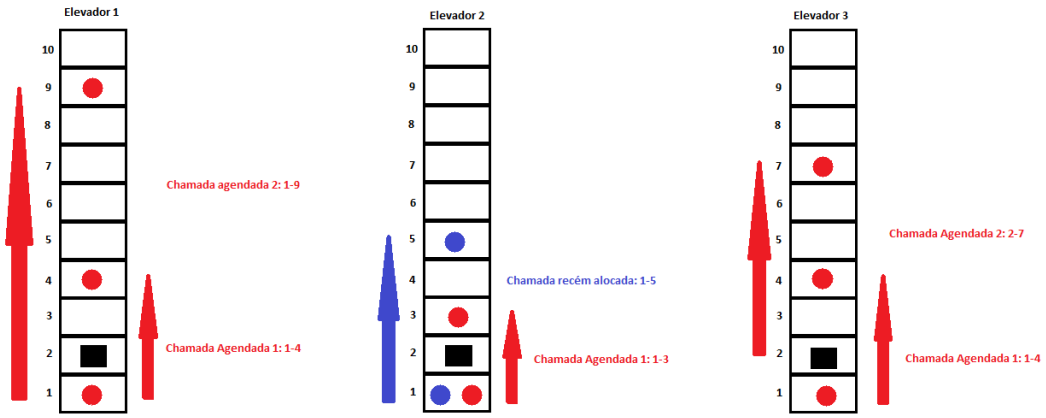


Figura 2.11: Situação de acúmulo de chamadas

Após todas essas análises de casos, a equação de custo que irá reger todo o sistema possuirá ambos elementos apresentados, tempo de espera e viajem, sendo ela uma combinação linear destes valores, como demonstrado abaixo:

$$f(i) = k_1 JT_i + k_2 WT_i \tag{2.21}$$

Onde cada parâmetro é definido como:

$\mathbf{JT}_i$: Tempo desde a entrada do passageiro no carro, até a abertura de portas no seu destino com o elevador i.

$\mathbf{WT}_i$: Tempo que o elevador i leva para chegar à origem.

$\mathbf{k}_n$:Fator de ponderação.

2.5 PROGRAMAÇÃO DINÂMICA

A pesquisa operacional é uma área de estudo que busca a alocação de recursos afim de encontrar a melhor solução (ponto ótimo) e para isso existem diversas ferramentas, dentre eles estão o método Simplex, caminho crítico e fluxo máximo. Existe ainda a programação dinâmica que é usada para criar uma sucessão de decisões inter-relacionadas, característico do sistema de elevadores porque quando um elevador não pode atender a uma chamada, algum outro deve atendê-la, portanto a decisão em um carro afeta todos os outros. A programação dinâmica, diferentemente da linear, não possui uma formulação matemática do seu problema, sendo necessário criar a equação particular da situação trabalhada.

2.5.1 CARACTERÍSTICAS DE UM PROBLEMA DE PROGRAMAÇÃO DINÂMICA

Existem aspectos básicos dos problemas de programação dinâmica que podem ser percebidos, de acordo com Hillier (2006) são eles:

- O problema pode ser dividido em estágios, nos quais uma decisão sobre a política a ser adotada é necessária a cada estágio.
- Cada estágio possui um número de estados associados ao início desse estágio.
- O efeito da decisão sobre a política a ser adotada a cada estágio é o de transformar o estado atual em um estado associado ao início do estágio seguinte (possivelmente de acordo com uma distribuição probabilística).
- O procedimento de resolução é desenhado para encontrar uma política ótima para o problema como um todo, isto é, estender a fórmula de decisão sobre a política ótima em cada estágio para cada um dos estados possíveis.
- Dado o estado atual, uma política ótima para os estágios restantes é independente das decisões sobre as políticas adotadas nos estágios anteriores. Portanto, a decisão imediata ótima depende somente do estado atual e não de como se chegou lá. Esse é o princípio da otimalidade para a programação dinâmica.
- O procedimento de resolução começa encontrando a política ótima para o último estágio.
- Há uma relação recursiva que identifica a política ótima para o estágio n, dada a política ótima para o estágio n + 1.

A Figura 2.12 exemplifica o tipo de problema que possui estados (A, B, C...), seus respectivos custos de deslocamento de um ao outro e também é possível ver a presença das características citadas acima. Uma possibilidade para resolver esse problema seria a tentativa e erro, todavia isso requer um grande esforço computacional para sistemas mais complexos e ainda pode acontecer de

ser impossível de encontrar todas as alternativas. A proposta da programação dinâmica é encontrar a solução ótima para um problema pequeno e a partir dela ir expandindo o problema em busca da nova solução ótima.

A Figura 2.12 é baseada no problema da diligência, onde um homem deseja ir do estado A ao J com os custos de translado mostrados na imagem. Então para resolver é preciso partir do destino (estado J) e analisando os trajetos disponíveis antes dele, ou seja, vindo do estado H ou I. Escolhida a melhor opção, recua-se mais uma etapa, trajetos vindos dos estados E, F e G, este padrão repete-se até chegar na origem, ponto A, resultando na solução ótima global, a qual foi baseada nas soluções ótimas parciais encontradas em cada etapa.

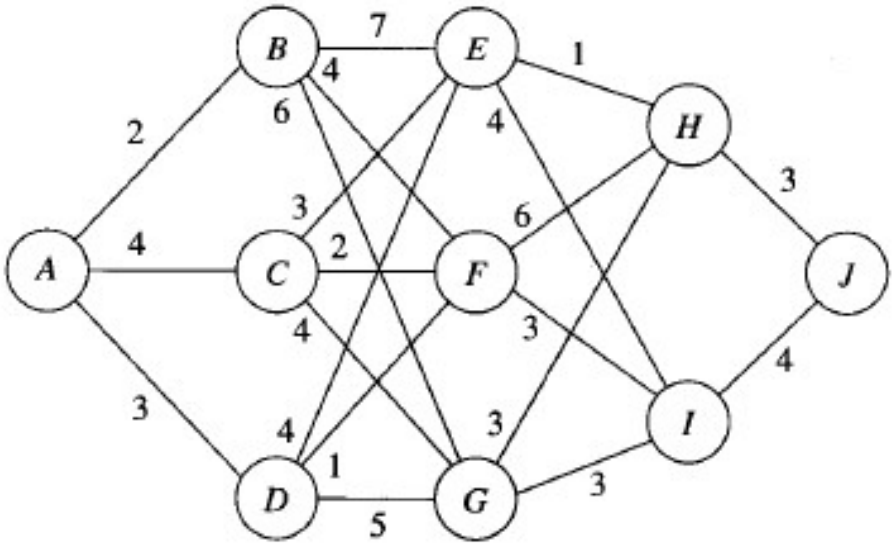


Figura 2.12: Sistema de rotas e custos (Hillier, 2006)

2.5.2 PROGRAMAÇÃO DINÂMICA DETERMINÍSTICA

Esta metologia é usada quando o estado do estágio seguinte é determinado completamente pelo estado e decisão atual. A estrutura básica desse tipo de programação está mostrado na Figura 2.13 onde n é o estágio, s é o estado atual, f é o valor da função custo e f* é a solução ótima para a função custo na próxima etapa.

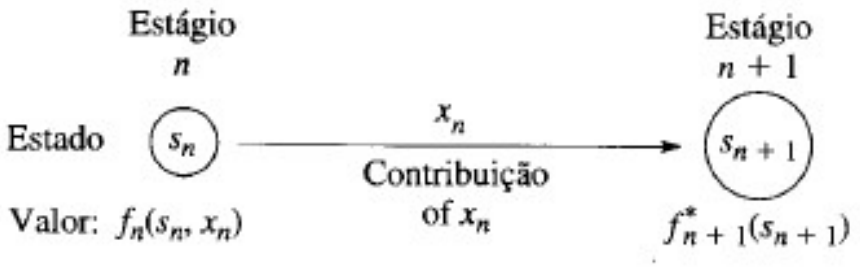


Figura 2.13: Programação dinâmica determinística (Hillier, 2006)

2.5.3 PROGRAMAÇÃO DINÂMICA PROBABILÍSTICA

Em oposição ao item anterior, na programação dinâmica probabilística o próximo estado não pode ser completamente determinado pelo estado e decisão atual, entretanto existe uma distribuição probabilística indicando o estado seguinte que é definido pelo estado e decisão atual. A Figura 2.14 mostra o diagrama de bloco com a estrutura deste tipo de programação, sendo que cada parâmetro mantém o significado da Figura 2.13.

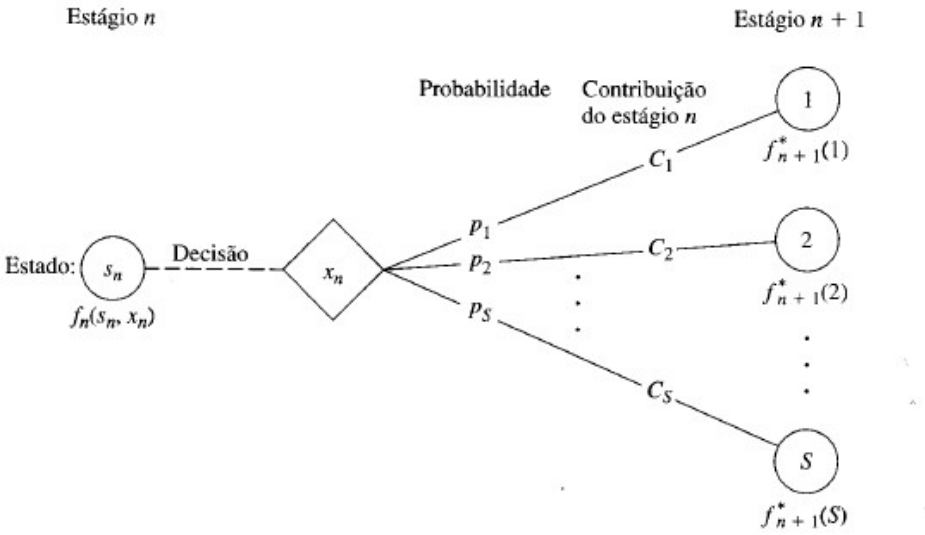


Figura 2.14: Programação dinâmica probabilística (Hillier, 2006)

2.6 SOFTWARES UTILIZADOS

2.6.1 ARENA

Este software foi escolhido devido a facilidade em gerar ambientes de simulação de eventos discretos e também em virtude de possuir uma interface bastante gráfica. Outra vantagem é a possibilidade de adquiri-lo com a licença estudantil, apesar de que esta versão é limitada, outro fator é que este programa é conhecido no meio acadêmico, tornando-se mais fácil encontrar conteúdo sobre ele.

Distribuído pela *Rockwell Automation*, o *Arena* possibilita a simulação de processos por meio de modelos matemáticos que proporcionam soluções analíticas. A parte gráfica é baseada na linguagem *SIMAN* (Arena, 2004), a Figura 2.15 apresenta um linha de produção sendo simulada através dos blocos na parte inferior e na parte superior a representação gráfica do sistema.

Estão presentes blocos de origem, destino decisão, atrasos... Que permitem reproduzir diversos cenários desde tráfego em um cruzamento de ruas até uma linha de produção. Enquanto simula, é mostrada a presença, se houver, de filas antes dos blocos, enquanto que ao final é apresentado um relatório contendo diversas informações como taxa de utilização de um determinado bloco,

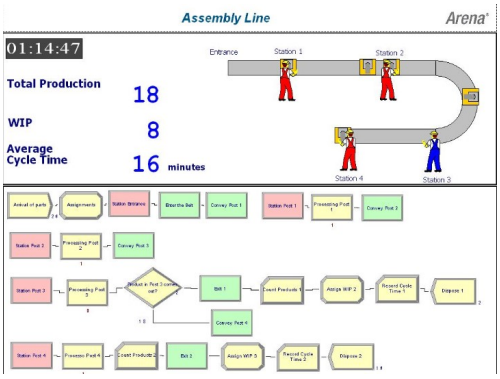


Figura 2.15: Simulação no Arena

tempo de espera na fila, tempo de processamento, unidades processadas... Estes dados podem ser vistos na Figura 2.16 que mostra tempos de espera, processamento, atendimento dentre outros para *Entity 1*.

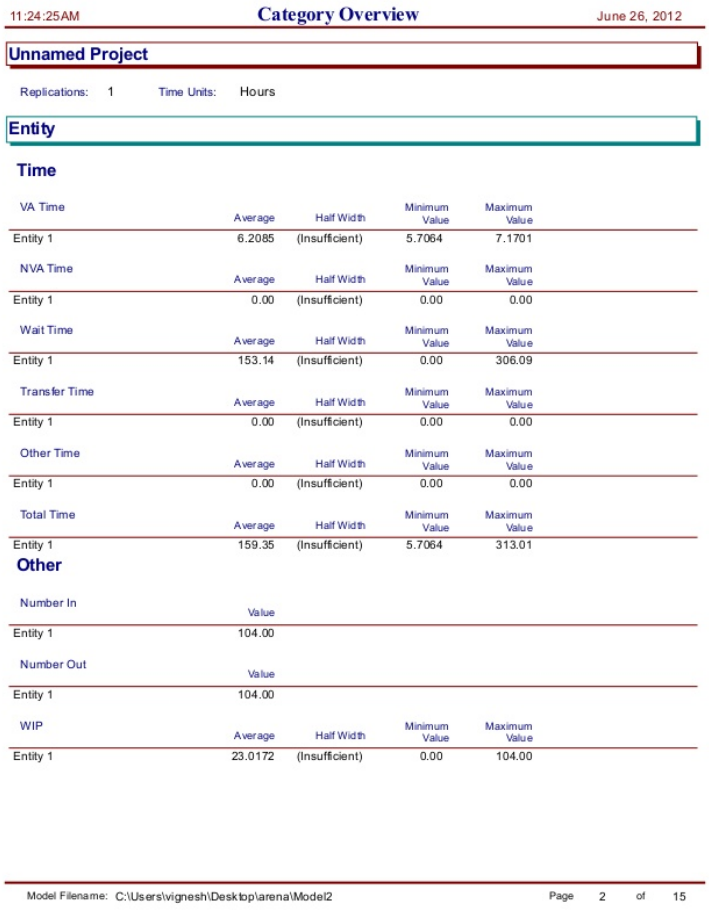
A simulação de processos é eficaz na tomada de decisões pois permite criar e testar modelos possibilitando assim predizer o que irá acontecer sem a necessidade de alterar o ambiente físico. Os principais passos para a modelagem são:

- Criar o modelo básico
- Refinar
- Simular
- Analisar os resultados da simulação
- Selecionar a melhor opção

2.6.2 VISUAL BASIC FOR APPLICATIONS

É uma aplicação do Visual Basic para os programas da Microsoft que permite ao usuário utilizar recursos de programação em documentos do Microsoft Officie como Word, Excel, Power Point... De maneira geral, o Visual Basic é um aperfeiçoamento do Basic (linguagem de programação) e do Visual(Pacote EX: Visual Studio), possuindo um ambiente para desenvolvimento integrado (IDE) totalmente gráfico, o que facilita muito a criação deste tipo de interface.

O VBA é bastante recomendado para automatizar tarefas, extensões para interação com o usuário e interação entre aplicativos do Office (Introdução ao VBA no Office, 2019). Para este trabalho a principal vantagem é a última tendo em vista que o Arena já possui o seu Visual Basic que é capaz de interagir com o Excel, onde ficam gravados os dados da simulação.



11:24:25AM **Category Overview** June 26, 2012

Unnamed Project

Replications: 1 Time Units: Hours

Entity

Time

VA Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	6.2085	(Insufficient)	5.7064	7.1701

NVA Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00

Wait Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	153.14	(Insufficient)	0.00	306.09

Transfer Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00

Other Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00

Total Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	159.35	(Insufficient)	5.7064	313.01

Other

Number In	Value
Entity 1	104.00

Number Out	Value
Entity 1	104.00

WIP	Average	Half Width	Minimum Value	Maximum Value
Entity 1	23.0172	(Insufficient)	0.00	104.00

Model Filename: C:\Users\vignesh\Desktop\arena\Model2 Page 2 of 15

Figura 2.16: Relatório da simulação no Arena

2.6.3 EXCEL

Criado em 1987, o Excel atualmente é um dos softwares da Microsoft mais utilizado, principalmente em empresas tendo em vista suas diversas funcionalidades relacionadas a manipulação de planilhas e gráficos.

Organizado em células em linhas e colunas, o Excel permite entender intuitivamente as operações que ocorrem em seu escopo, tendo em vista que é muito usado para performar cálculos simples, mas muito repetitivos. Além desta aplicação, é um ótimo local para organizar e controlar dados, como por exemplo despesas, folhas de pagamentos, controle de estoque... No projeto apresentado, o Excel está sendo utilizado para organizar e armazenas os dados da simulação que serão constantemente alterados pelos elevadores e processados pelo otimizador.

Capítulo 3

Metodologia

A realização do trabalho está divida em duas etapas principais: (1) Escolher o elevador que possui o menor custo para atender a chamada por meio do otimizador feito em VBA (2) simular o sistema utilizando o *Arena*, esta parte ainda pode ser separada no gerador de chamadas, que está diretamente relacionado com o movimento dos passageiros, e a trajetória dos elevadores que simula cada andar do prédio e todos os atrasos que podem ocorrer.

3.1 PROBLEMA DA DECISÃO

Primeiramente é necessário entender como os estados, custos e iterações estão sendo interpretadas neste trabalho. Ao surgir uma nova chamada, esta pode ser atendida por qualquer um dos elevadores, entretanto cada um possuirá seu respectivo custo para fazê-lo. Na Figura 3.1 é possível ver o sistema, então considerando que o conjunto esteja na iteração N e no estado A quando uma nova solicitação é criada e ela por ser atendida pelo elevador 1 (Transição A-B), pelo elevador 2 (transição A-C) ou pelo 3 (Transição A-D), cada qual com seu respectivo custo, este valor dependerá dos parâmetros dos sistema neste instante N. Decido o carro, o sistema passa para a iteração N+1, portanto cada mudança é causada pelas novas chamadas.

A ideia central da abordagem utilizada neste trabalho é encontrar o menor custo de sair de um estado atual para o próximo, o que deriva um pouco do caminho crítico, entretanto a diferença principal é que não se tem acesso aos custos dos estados seguintes já que não é possível saber as origens e destinos dos futuros passageiros antes que eles registrem sua chamada no DCS.

Como já citado anteriormente os tempos de espera do passageiro até o elevador chegar(WT) e também de viagem(JT) serão considerados diretamente nestas decisões, já que fazem parte da função custo. Suas definições já foram mostradas anteriormente, entretanto estes assuntos serão retomados na seção de otimização pelo VBA. Em resumo a escolha será baseada no custo atual que uma chamada terá para ser alocada em cada um dos elevadores considerando seus dados como direção, andar atual, chamadas na memória entre outros, portanto é uma decisão tomada para um determinado instante N, o qual é o ponto ótimo local, entretanto não é possível avaliá-lo globalmente pois apenas os dados dos estados anteriores estão disponíveis para o sistema.

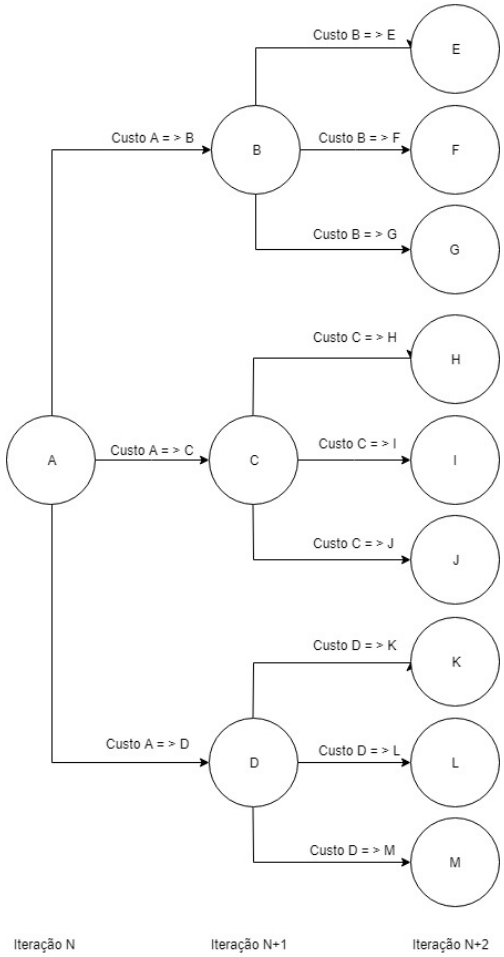


Figura 3.1: Representação dos estados para a alocação de chamadas

3.2 CRIANDO O MODELO NO ARENA

O gerador de chamadas utilizado foi uma versão muito semelhante ao do trabalho do Rodriguez (2015) o qual está mostrado na Figura 3.2. Infelizmente, por limitações do Arena, não foi possível obter imagens em melhor resolução, entretanto os blocos são descritos da esquerda para a direita. Os passageiros são representados pela entidade que irá percorrer os blocos, inicialmente ela é criada em "Chegada"e o bloco de decisão "Padrão Uppeak"garante que a maioria das chamadas tenham origem no térreo fazendo que em 90% dos casos o bloco *Assign* "Origem Terreo"seja selecionado, este atribui como origem o andar 1, os 10% restantes partem de quaisquer outros andares atribuídos pelo bloco "Outras Origens", a seguir o bloco "Destino"é responsável por escolher para onde cada usurário deseja ir.

Neste ponto cada passageiro (entidade) já possui sua respectiva origem e destino (atributos), o bloco "Descartando Chamadas"é responsável por retirar as solicitações onde o andar de origem é o mesmo do andar de destino já que não faz sentido alguém chamar um elevador para o andar onde ela já está. Logo a seguir há um bloco *Assign* que conta a quantidade de passageiros no sistema e então o bloco VBA que contém o otimizador o qual informará o elevador selecionado, dado utilizado no último bloco de decisão da imagem. A chegada dos passageiros no andar principal obedecerá uma

distribuição de Poisson pois foi a abordagem usada nos trabalhos antigos (Rodriguez, 2015) (Alvaro, 2010)e recomendada pela Gina Barney (2003). Esta distribuição possui a equação mostrada logo abaixo, onde P é a probabilidade de chegada de alguém dentro de um tempo T, sendo a taxa de chegada λ e o n é o número de eventos chegada de passageiros.

$$P\left(n,T\right)=\frac{e^{-(\lambda T)}(\lambda T)^{n}}{n!},n=1,2,3... \tag{3.1}$$

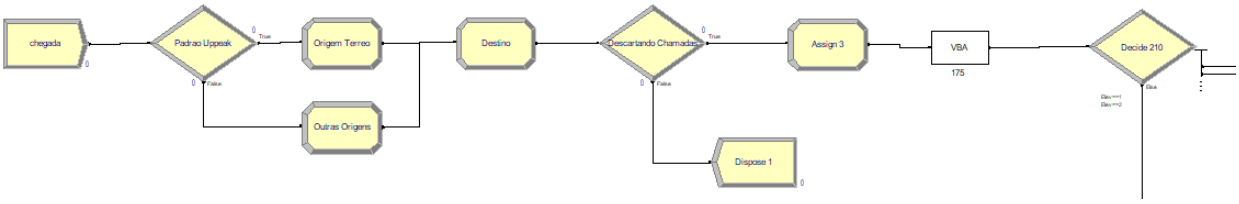


Figura 3.2: Gerador de chamadas do Arena

A taxa λ é encontrada a partir equação descrita a seguir, sendo TI(%) a intensidade do tráfego e POP_i a quantidade de pessoas em cada andar, portanto o somatório é o valor total presente no prédio.

$$\lambda=\frac{0,01(TI)}{300}\sum_{i}POP_i \tag{3.2}$$

Após cada pessoa ser designada para seu respectivo elevador, ela deve esperá-lo chegar em seu andar de origem para embarcar, depois é necessário que ela espere até o carro chegar no andar de destino. Estas esperas estão representadas na Figura 3.3 onde dentro do retângulo azul cada bloco Hold mantém a entidade até que o andar atual do seu elevador seja igual ao seu andar de origem. A outra coluna de Hold's também retém a entidade até o momento em que o andar de destino seja o mesmo do andar atual. Passadas todos essas esperas, a pessoa segue para seu local de destino, representado pelos blocos "*Dispose*". O andar atual de cada carro é informado e constantemente atualizado pela trajetória dos elevadores, dinâmica que será explicada a seguir.

Até o momento foi tratada a parte referente à movimentação dos passageiros, todavia é preciso simular o movimento dos elevadores pois eles estarão constantemente alterando as variáveis do sistema, sendo necessário calcular um novo custo a cada nova chamada. A Figura 3.4 apresenta, de maneira geral, como foi feita a trajetória dos elevadores no prédio onde é possível ver três partes idênticas para cada um dos elevadores. A parte inicial pode ser vista na Figura 3.5, o bloco *Create* gera a única entidade Elevador para esse sistema, ela seguirá para o bloco de VBA o qual lê os dados atuais do sistema que estão armazenados em *Excel*, como o andar atual do elevador e a próxima parada agendada. Se não houver mais solicitações para ele, o elevador fica retido no *Hold*. É importante notar que não há *Dispose*, portanto a entidade ficaria se movendo continuamente, entretanto esse *Hold* garante que o elevador terá que parar e esperar a próxima solicitação caso não haja nenhuma agendada, senão ele seguirá para "Andar_Agora2".

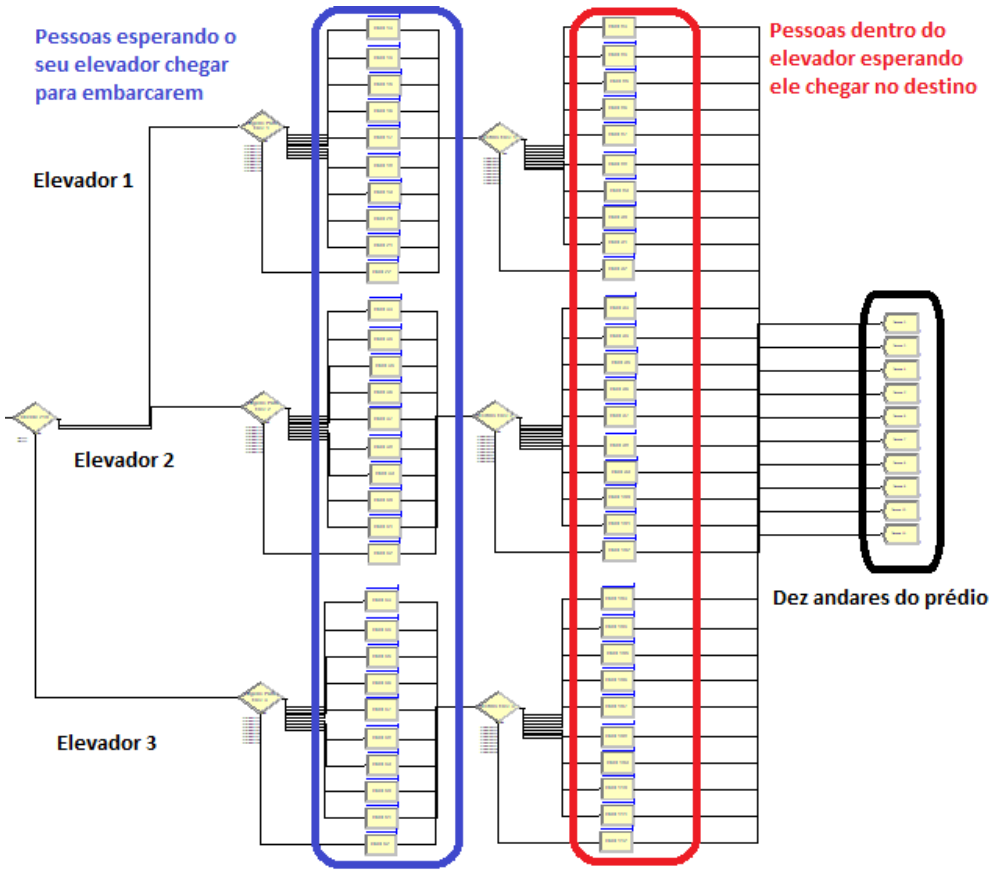


Figura 3.3: Esperas dos passageiros

Por fim os andares estão presentes na Figura 3.6 onde o primeiro *Decide* verifica se o elevador estava previamente parado pois neste caso é necessário acrescentar um tempo de aceleração, também é interessante notar que os blocos com "P"no início são os reservados para este evento. Logo a seguir é examinado se o este andar é onde o elevador precisa parar, se por ventura não for, ele segue para subindo ou descendo atualizando as informações de direção e andar atual do elevador. Caso a próxima parada seja de fato neste andar primeiramente é sinalizado que o elevador está apto a entrada ou saída de passageiros e atualiza o andar atual. Após um certo tempo ele fecha a porta e começa a movimentar novamente, "Setando_Partida"garante que não é mais possível entrar no carro, portanto os próximo usuários terão de esperá-lo. O segundo VBA desconta a parada que fora atendida e renova as demais informações no *Excel*, por fim existem os atrasos relacionados aos tempos de movimentação como aceleração por exemplo.

3.3 OTIMIZAÇÃO PELO VBA

O Arena possui uma versão do *Visual Basic* que é utilizado para ler os dado *Excel*, o qual será o banco de dados de todas as informações do sistema, processá-los e então salvá-los novamente no *Excel* para futuras iterações. Um ponto relevante é entender que existem três tipos de dados neste projeto, o primeiro são os que estão salvos no *Excel*, como mostrado na Figura 3.7 (estas são

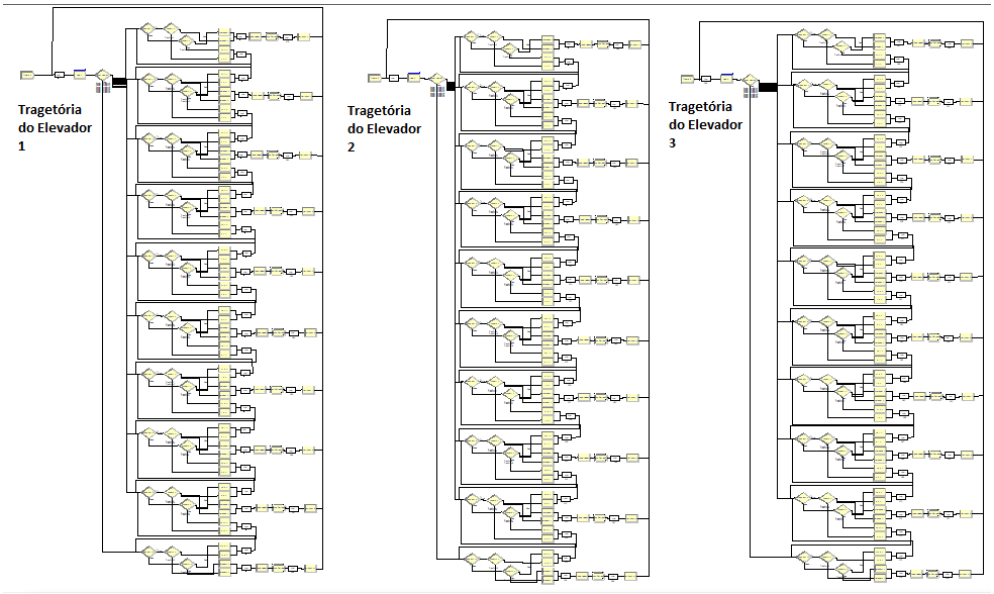


Figura 3.4: Movimentação dos três elevadores

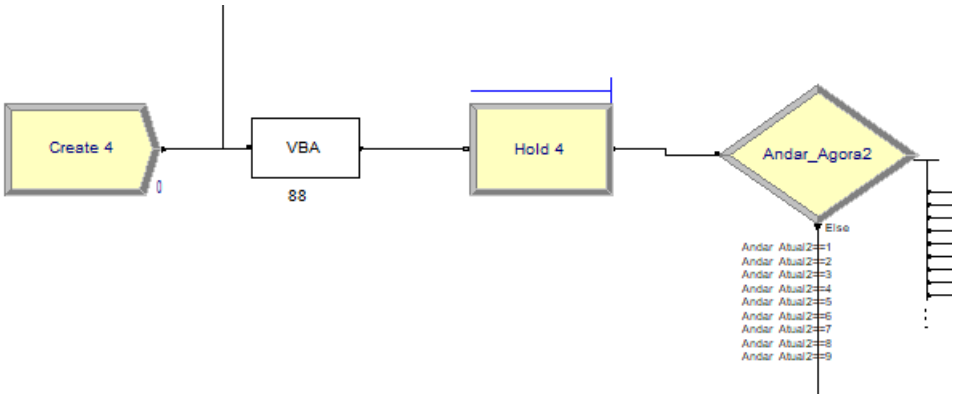


Figura 3.5: Início da trajetória do elevador

informações meramente ilustrativas, não estão otimizadas ou com os custos corretos), o segundo conjunto são as variáveis, atributos e entidades que irão comandar os blocos do Arena e em terceiro são variáveis locais dentro do código VBA usadas para receberem os valores da simulação e do *Excel* para otimizar o sistema. Alguns dos dados lidos são: andar atual, direção, número de paradas agendadas, seus respectivos andares e seus custos.

Com as informações coletadas é possível iniciar os cálculos do tempo que cada elevador vai demorar até chegar na origem (WT), mas para isso é preciso encontrar a melhor forma de encaixar a nova chamada na lista das que já estão pendentes, isto foi feito utilizando a lógica apresentada tópico sobre a função custo do elevador. Concluída esta etapa, respectivamente é calculado o tempo de viagem (JT) encontrando o melhor local para alocar o destino da chamada e o custo até alcançá-lo considerando todas as paradas realizadas no trajeto. O resultado de cada elevador será a combinação linear destes dois elementos, por fim o melhor carro para a nova solicitação é aquele com o menor custo, este terá a nova chamada inserida na sua memória e o número de paradas atualizado no *Excel* e também na simulação do Arena. Como mencionado anteriormente,

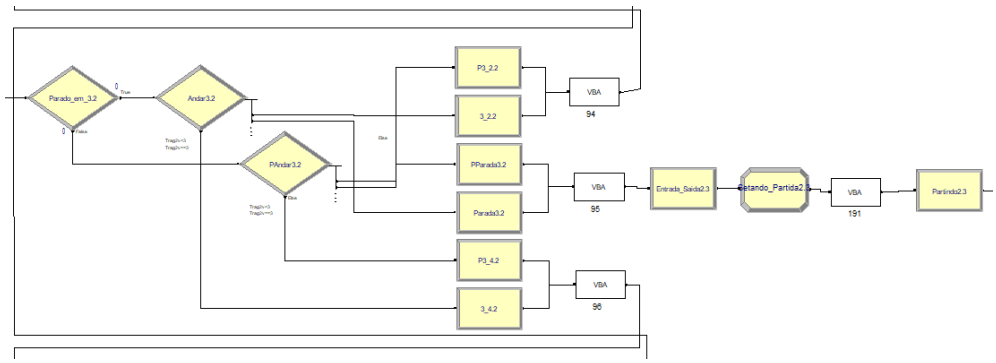


Figura 3.6: Simulação de um andar

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	6	Quantidade de			4	1	5	10	1	3	Paradas alocadas do elevador 1			
2	7	paradas agendadas			14	27	45	74	89	104	Custos das paradas do elevador 1			
3	5	para cada elevador												
4					3	7	9	4	6	2	5	Paradas do Elevador 2		
5	Estado inicial		Andar atual		21	28	36	42	59	68	84	Custos das paradas		
6	0	4	1											
7	0	3	6		10	1	4	9	2	Paradas do elevador 3				
8	0	4	9		5	18	36	45	63	Custos das paradas				
9		Quantidade de												
10		pessoas												
11														
12														
13														

Figura 3.7: Dados salvos no *Excel*

a quantidade de pessoas no elevador será contabilizada apenas para evitar a excedência de carga no carro, e para isso foram feitas algumas modificações, como é possível ver a presença de novos blocos VBA's Figura 3.8 que descontam o número de pessoas atualmente no elevador toda vez que uma entidade (passageiro) deixa o sistema.

Afim de usar a melhor combinação linear entre os parâmetros da função de transferência, foram feitos diversos testes para verificar qual possuiria o menor tempo de saída médio (tempo desde a chegada do passageiro na origem até seu destino) em uma situação de demanda crescente até o máximo de 16% da população em cinco minutos. O resultado está na Figura 3.9, sendo escolhido k=0,9 já que reduz em aproximadamente 3 segundos o tempo de saída em relação aos que ficaram em segundo, k=0,1, k=0,2 e k=0,3. Portanto a função custo utilizada neste trabalho pode ser vista logo abaixo;

$$f(i) = 0,1JT_i + 0,9WT_i \tag{3.3}$$

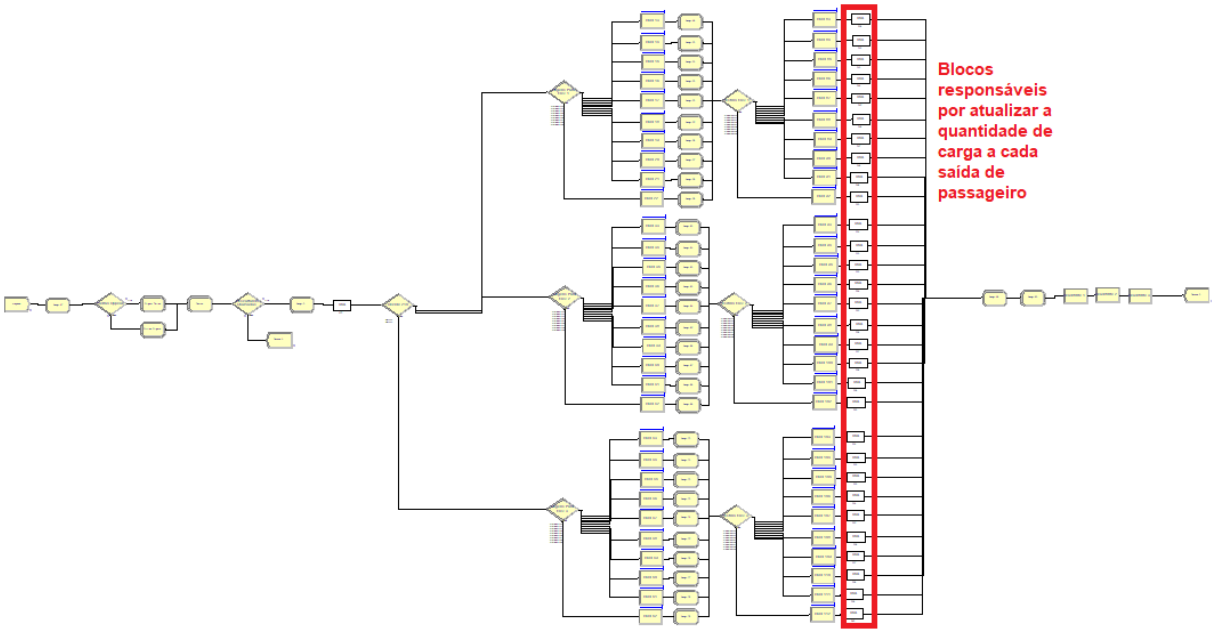


Figura 3.8: Modificações para atualizar informação sobre a carga

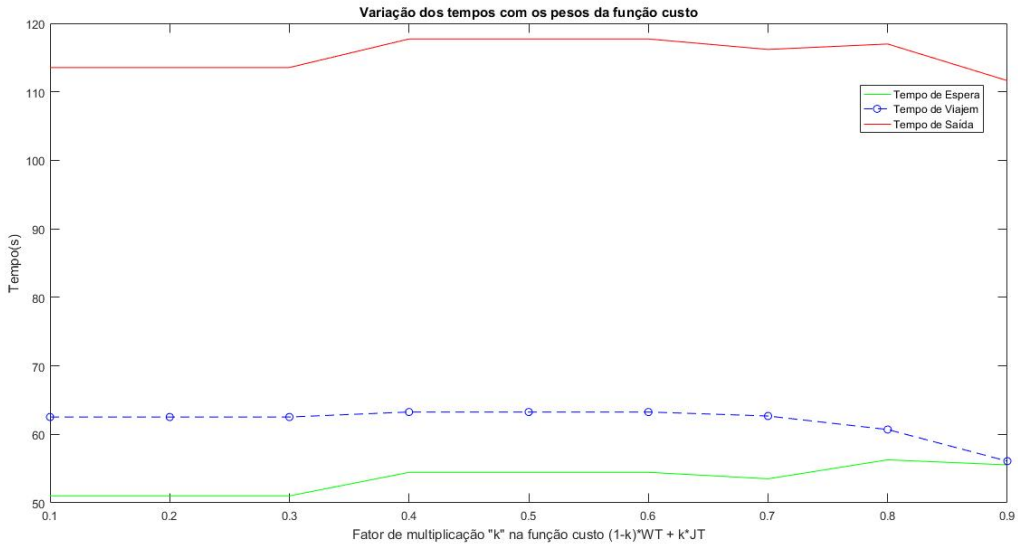


Figura 3.9: Influência das ponderações da função custo no tempo de saída

Capítulo 4

DISCUSSÃO DE RESULTADOS

Os testes foram feitos para diferentes quantidades de pessoas no sistema, até o limite de 16% da população do prédio. Para os cálculos desses tempos foi necessário fazer algumas modificações no projeto original afim de coletar tais dados, tais mudanças estão mostrados na Figura 4.1. As equações abaixo indicam como os cálculos dos tempos foram feitos para o i-ésimo passageiro no sistema.

$$Tempo\_Espera_i = Tempo\_Fila_i - Tempo\_Chegada_i \tag{4.1}$$

$$Tempo\_Viajem_i = Tempo\_Saida_i - Tempo\_Fila_i \tag{4.2}$$

$$Tempo\_Destino_i = Tempo\_Saida_i - Tempo\_Chegada_i \tag{4.3}$$

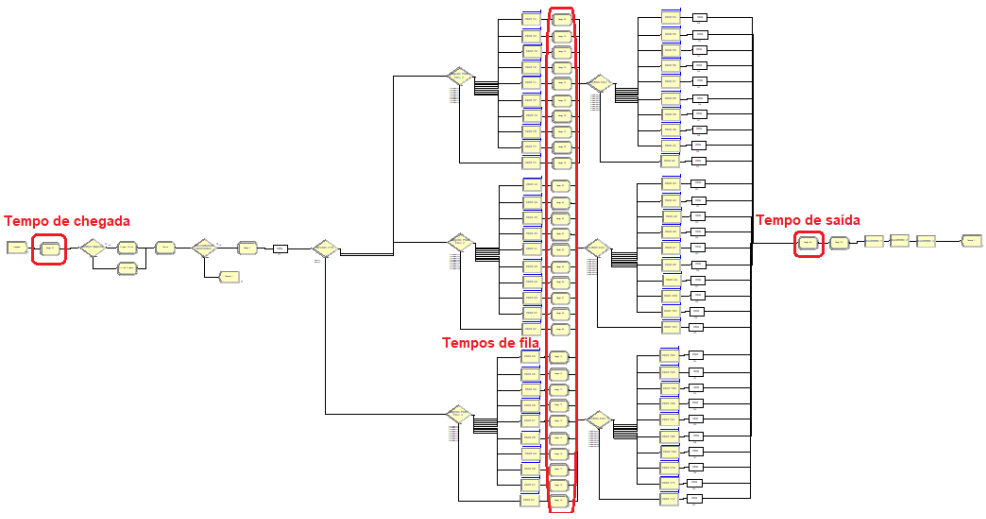


Figura 4.1: Modificações feitas no percurso do passageiro para coletar tempos

Tabela 4.1: Resultados dos tempos para diferentes demandas

Porcentagem da População	**Quantidade de Passageiros**	**Tempo médio de espera (s)**	**Tempo médio de voo (s)**	**Tempo médio de destino (s)**
2	9	16,88	31,2	48,08
4	18	40,87	37,7	78,56
6	27	47,35	44,82	94,18
8	36	51,25	51,24	102,77
10	46	55,54	56,09	111,63
12	55	62,48	57,035	119,51
14	64	68,67	63,09	131,76
16	73	69,27	67,41	136,68

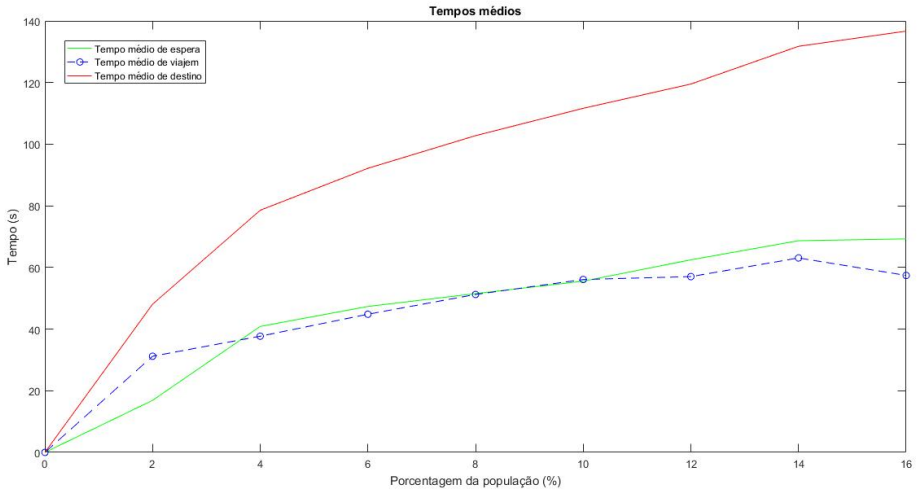


Figura 4.2: Tempos médios

Para uma melhor visualização, as informações da Tabela 4.1 foram plotados na Figura 4.2. Percebe-se que existem três momentos diferentes, primeiro quando de 0% a 4% da população está utilizando o sistema e a média tempo de espera é inferior a de viajem isto ocorre pois com poucos passageiros é mais provável que o elevador consiga atender a demanda em uma única viajem, sem precisar voltar para depois atender algum usuário deixado para trás. Devido aos carros estarem inicialmente no térreo e 90% das chamadas terem este andar como origem, o atendimento destes primeiros usuários é facilitado. Em contra partida, a quantidade de paradas no trajeto cresce mais e faz a o tempo de viajem ser maior, já que o elevador está atendendo a todos ou pelo menos à maioria das pessoas na primeira viajem.

A partir dos 4% até os 10% ambos os tempos tendem a se aproximar, portanto a pessoa ficará, em média, um mesmo período esperando pelo carro e dentro dele até chegar ao seu destino. Depois dos 10% o tempo de espera supera o de viajem pois com tantos passageiros chegando em cinco

minutos, os elevadores deixam um maior número de gente para trás, fazendo-se necessário fazer movimentações de ida e volta, acrescentando assim, o custo até chegar no andar de origem, como já foi explicado no capítulo sobre a função custo.

Um parâmetro interessante a ser analisado é a quantidade de paradas que os elevadores tiveram que realizar pois cada uma significa aceleração e desaceleração do carro, aumentando a energia gasta pelos motores. Portanto em um futuro estudo de eficiência energética, o qual não foi o foco neste trabalho, esses dados podem vir a ser úteis. A Figura 4.3 representa a informação contida na Tabela 4.2 mostrando que cada carro tem um número de paradas semelhante, portanto um consumo energético balanceado entre os três elevadores, considerando apenas gastos em aceleração e desaceleração, para uma análise mais profunda, também seria necessário a carga afim de averiguar os esforços de cada motor devido ao peso de cada carro. Outra maneira de ver esse parâmetro é globalmente, ou seja, o número total de paradas, buscando reduzi-lo como um todo, tais dados estão presentes na Figura 4.4.

Tabela 4.2: Número de paradas de cada Elevador

Porcentagem da População	**Quantidade de Passageiros**	**Elevador 1**	**Elevador 2**	**Elevador 3**
2	9	5	4	3
4	18	6	7	6
6	27	6	8	7
8	36	7	8	11
10	46	7	8	11
12	55	10	11	12
14	64	10	14	13
16	73	11	14	13

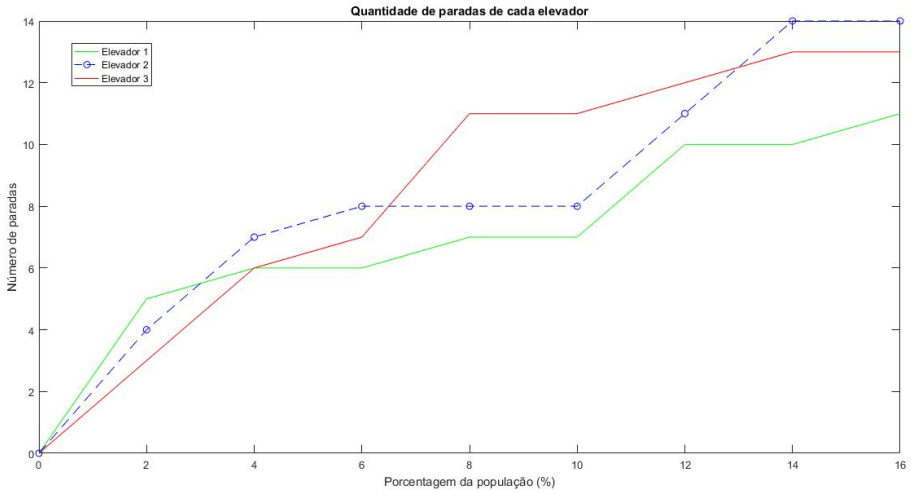


Figura 4.3: Quantidade de paradas de cada elevador

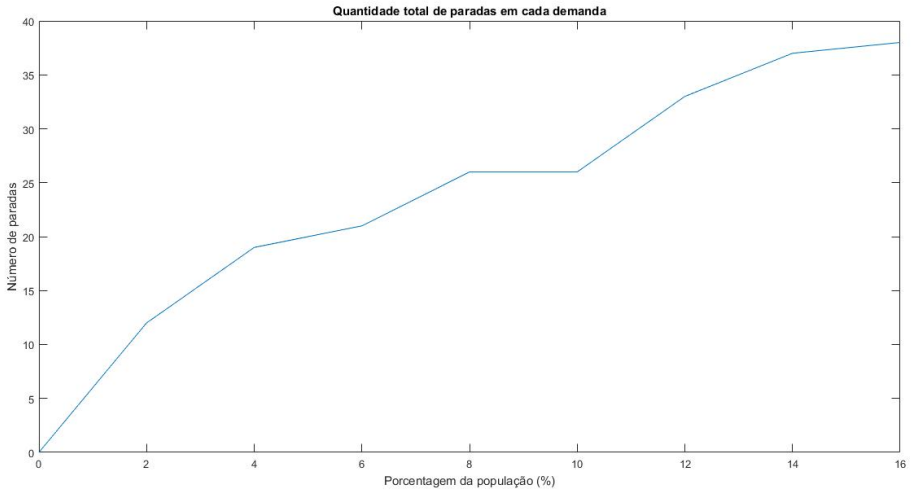


Figura 4.4: Quantidade total de paradas dos elevadores

4.1 COMPARAÇÃO DOS RESULTADOS COM OUTROS AUTORES

O principal objetivo deste trabalho é obter resultados para ser comparado com outras abordagens e trabalhos anteriores afim de tirar conclusões mais amplas em relação aos diferentes modos de se otimizar o problema do controle de elevadores. Todavia as comparações aqui feitas serão mais qualitativas porque não se tem acesso às condições iniciais utilizada de cada um e nem a filosofia de funcionamento dinâmico do elevador como por exemplo a contagem dos tempos. Em outras palavras, não é conhecido a quantidade de andares dos prédios, de elevadores, de pessoas e tão pouco os tempos de aceleração, abertura de portas, dentre outros.

A comparação mais fiel é com o trabalho de Rodriguez (2015) pois foi possível ter acesso aos dados citados anteriormente, todavia mesmo sabendo as condições iniciais, ainda podem existir diferenças na modelagem do sistema dinâmico que influenciam diretamente nos resultados finais. Os dados de ambos os trabalhos podem ser vistos nas Figuras 4.5, 4.6 e 4.7 que comparam os tempos de espera, viajem e destino, respectivamente.

O ponto mais evidente a ser citado é que em todas as demandas, os tempos do trabalho que usou algoritmos bioinspirados (Alvarez, 2010) foram menores , partindo das mesmas características do sistema. Este fato a princípio mostra uma ineficiência do método em relação ao utilizado pelo Rodriguez (2015), no entanto o problema pode não estar na otimização.

A Figura 4.8 apresenta a quantidade de paradas que os elevadores tiveram que efetuar durante a simulação de cinco minutos. Os valores são próximos, sendo que até os 10% o método proposto possui mais paradas, o que muda logo em seguida, mas o importante a ser apontado é que por ambos possuírem quantidades semelhantes, a utilização do sistema como um todo também está sendo similar. Isso leva a crer que a disparidade dos tempos vista anteriormente pode ter sido causada por diferentes modelagens do sistema de trajetória dos elevadores e/ou na contagem dos

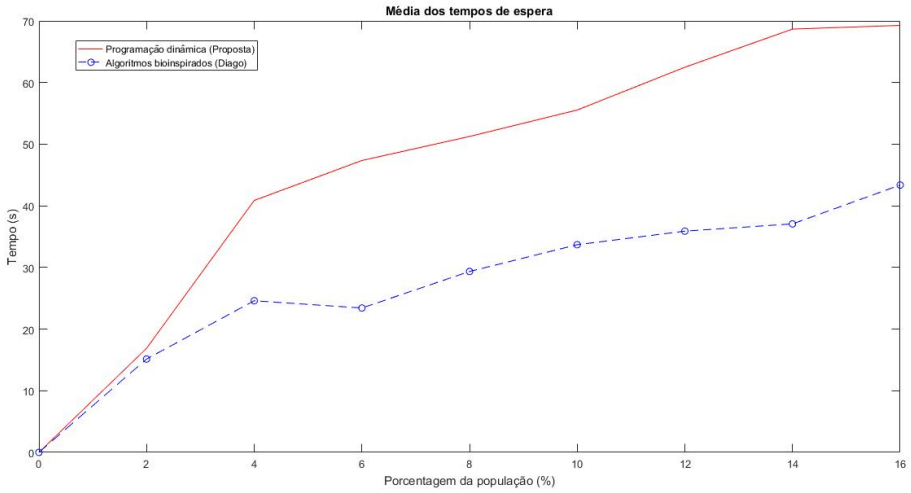


Figura 4.5: Comparação entre as médias dos tempos de espera

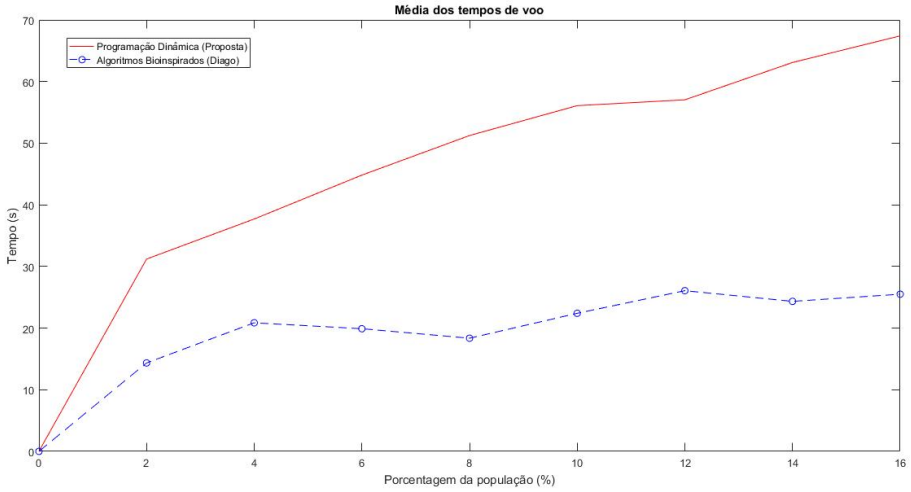


Figura 4.6: Comparação entre as médias dos tempos de viajem

tempos.

Outra possível causa para a diferença vista é na modelagem da trajetória em relação ao tempo de entrada e saída de passageiros (Tes) pois quando o carro parar em um andar, os usuários tem apenas esse intervalo para entrar e saírem, depois disso o elevador não está mais apto a receber novos usuários pois começará a se movimentar, então as próximas pessoas que chegarem, terão as suas chamadas agendadas posteriormente. Entretanto a contagem deste tempo deveria ser reinicializada a cada pessoa entrando ou saindo, isso permitiria uma estadia maior do carro no andar, possibilitando a uma maior quantidade de pessoas serem atendidas mais rapidamente. Além disto, não é conhecido se essa permissão de entrar e sair do carro no trabalho do Rodriguez (2015) foi tão rígida.

Por fim a Tabela 4.3 apresenta os valores obtidos usando outras abordagens. Devido aos

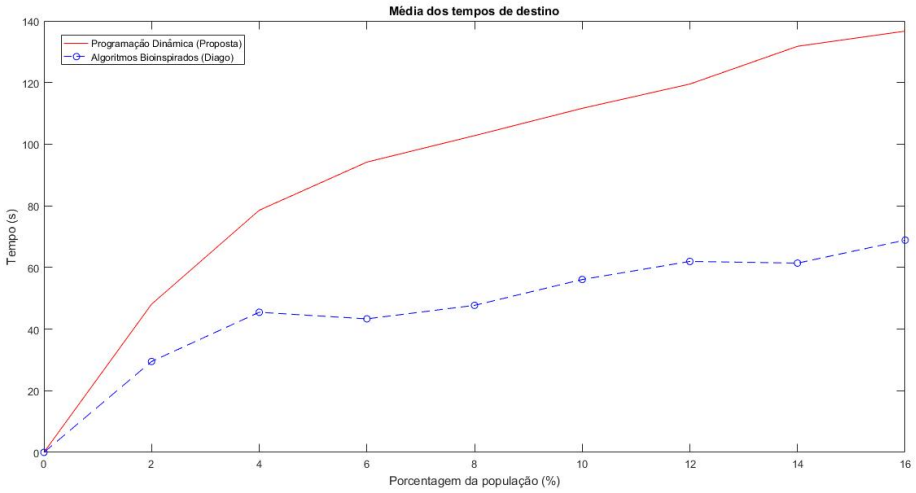


Figura 4.7: Comparação entre as médias dos tempos de destino

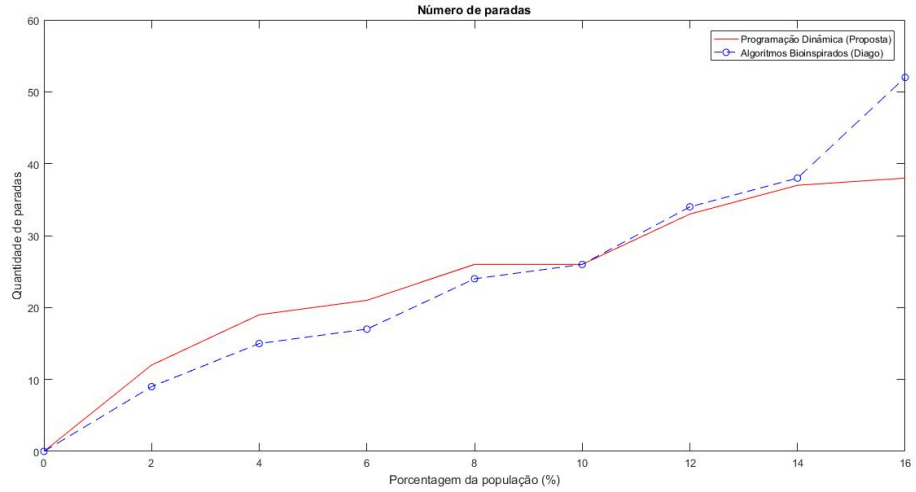


Figura 4.8: Comparação entre os números de paradas

problemas relatados anteriormente, a análise será qualitativa, portanto percebe-se que novamente os resultados foram piores que os demais autores, mesmo que em alguns casos com uma diferença não muito grande. Estas discrepâncias podem ser devido a diferentes condições iniciais ou formas de modelar ou ainda a ambos.

Tabela 4.3: Resultados de outros autores (Rodriguez, 2015)

Ano	**Autor**	**Trabalho**	**Tempo médio de espera**	**Tempo médio de voo**	**Tempo médio de destino**
2000	Siikonen	On traffic planning methodology	37	x	x
2004	Jian Liu	Dynamically dispatching method aiming to reduce the servicing time in the egcs	23,3	27,9	51,2
2010	Morkon	Design of elevator group control system simulation platform based on shortest distance algorithm	35,45	30,46	65,91
2011	Patiño	Fuzzy elevator group control system using technology for industrial automation	47	20	67
2011	Sheng	A Novel Elevator Group Control Scheduling Algorithm based on Pseudo Differential Feedback	35,85	15,83	51,68
2012	Y.Gu	Multi-objective optimization of multi-agent elevator group control system based on real-time particle swarm optimization algorithm	44,87	38,34	83,21
2013	Cortes	A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems	x	x	33
2015	Rodriguez	Otimização com algoritmos bioinspirados de controle de tráfego em grupo de elevadores	43,37	25,5	68,87
2019	Este Trabalho	Controle ótimo para grupo de elevadores usando programação dinâmica	64.59	75.2	139.8

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 CONCLUSÃO

Este trabalho se propôs a aplicar os princípios da programação dinâmica no problema do controle de elevadores, utilizando de softwares que auxiliassem nesta modelagem e simulação de um padrão uppeak. Durante as diversas simulações e testes feitos no otimizador, os custos e trajetórias estavam sendo calculadas corretamente, sempre alocando tanto a origem quanto a chamada o mais rápido possível, mas sempre obedecendo os critérios já explicados no capítulo sobre modelagem.

Entretanto os resultados adquiridos não foram satisfatórios em relação a literatura já existente desta área, pois mesmo com um trabalho que possui características mais semelhantes a este, que foi o do Rodriguez, os tempos foram superiores, indicando que ainda há melhorias a serem feitas tanto em relação a otimização quanto a modelagem.

A maior contribuição deste trabalho foi o desenvolvimento do simulador que em tempo real reproduz as trajetórias dos passageiros e dos carros, cada um deles agindo de maneira independente, onde cada pessoa deve esperar pelo seu respectivo carro chegar na sua origem e depois no destino.

5.2 TRABALHOS FUTUROS

Quanto ao otimizador é possível acrescentar outros parâmetros na função custo e encontrar uma outra combinação linear destes, como por exemplo o aumento dos tempos das viajem já alocadas na memória para uma chamada ser inserida no elevador, eficiência energética, quantidade de pessoas no carro, já que este é apenas utilizado como critério de desempate, entre outros.

Em relação a simulação pode-se alterar a contagem do tempo de entrada e saída das pessoas do carro, como já citado, mudar o arranjo dos tempos tornando o sistema como um todo mais próximo da realidade e também coletar mais informações sobre a simulação para avaliar os parâmetros sugeridos acima.

ANEXOS

I. REFERÊNCIAS BIBLIOGRÁFICAS

ALVARO Patiño Forero, Estudo e simulação de técnicas de controle de tráfego de grupo de elevadores usando automação industrial, 2010, Tese de mestrado, Universidade de Brasília, Brasil.

ARENA, Rockwell User Guide, 2004, p.2.

BARNEY, G. Elevator Traffic Handbook: Theory and practice. [S.l.]: Taylor & Francis, 2003.

Berna Bolata, Oguz Altunb, and Pablo Cortés, "A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems,"Applied Soft Computing 13 , 2013.

David L. Pepyne and C.G. Cassandras, Optimal Dispatching Control for Elevator Systems During Up Peak Traffic, Dezembro 1997, IEEE Transactions on control system technology, vol 5. EBERHART and KENNEDY, A new optimizer using particle swarm theory MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 06 agosto 2002 IEEE Xplore

HILLIER, L. Introducao a Pesquisa Operacional. [S.l.]: McGraw Hill, 2006, ed 8, p. 424-455

HISTORY of Elevators. Disponível em: http://www.elevatorhistory.net/, 2018, acesso em 10/10/2018.

INTRODUCAO ao VBA no Office, 01 de janeiro de 2019. Disponível em: https://www.microsoft.com/pt-br/, Acesso em 17/05/2019

LI, L. Applying Approximate Dynamic Programming to the Elevator Dispatching Problem, 2014, University of Queensland.

RODRIGUEZ, J. P. D. Otimização de controle de tráfego em grupo de elevadores com algoritmos bioinspirado, 2015, Dissertação de Mestrado ? Universidade de Brasília.

SIIKONEN, M.-L. On traffic planning methodology. International Congress on Vertical Tranportation, Elevcon Berlin 2000, Berlin, Alemanha, 2000.

SHENG Wu and GUIFANG Wu, "A Novel Elevator Group Control Scheduling Algorithm based on Pseudo Differential Feedback,"Proceeding of the IEEE, International Conference on Automation and Logistics, 2012.

WANG Chuansheng and CHEN Chunping, "Design of Elevator Group Control System Simulation Platform Based on Shortest Distance Algorithm,"International Conference on Electrical and Control Engineering, 2010.

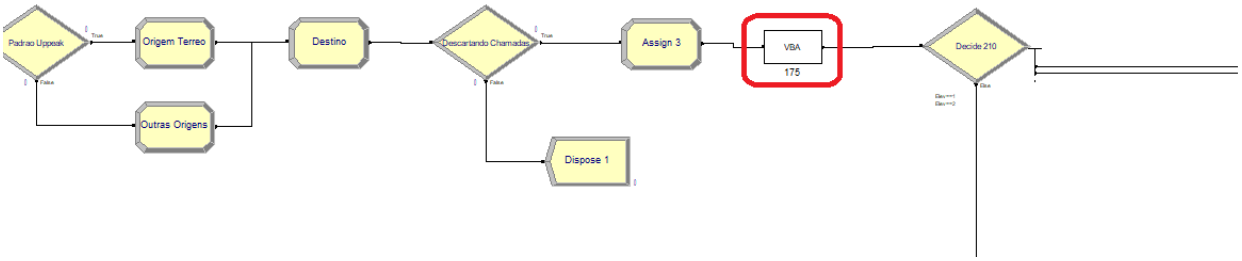
WESSELOWSKI, K.S., C.G. CASSANDRAS. 2006. The elevator dispatching problem: Hybrid system modeling and receding horizon control. C. Cassandras, A. Giua, C. Seatzu, J. Zaytoon, eds., Proceedings of 2nd IFAC Conference on Analysis and Design of Hybrid Systems. Elsevier, Alghero, Itália.

YANWU Gu, "Multi-Objective Optimization of Multi-Agent Elevator Group Control System

Based on Real-Time Particle Swarm Optimization Algorithm,"Engineering, Vol. 4 No. 7, 2012.

ZHOU and YE, Y. Q. Dynamically dispatching method aiming to reduce the servicing time in the EGCS, IEEE, 18 outubro de 2004, Hangzhou, China.

I. DESCRIÇÃO DO CONTEÚDO DO CD




```
Private Sub VBA_Block_175_Fire()
2


Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN
Dim Dest As Integer
Dim Orig As Integer
Dim i As Integer
Dim DIR1 As Integer
Dim DIR2 As Integer
Dim DIR3 As Integer
Dim Paradas As Integer
Dim Paradas_acima As Integer    'Quando a chamada é descendente, conta o número de paradas acima da origem
Dim Ta As Integer               'Tempo de viagem entre os andares
Dim Tac As Integer              'Tempo de aceleração ou desaceleração
Dim Tp  As Integer              'Tempo de abertura e fechamento de portas
Dim Tes As Integer              'Tempo de entrada ou saída de pessoas
Dim Ultimo_andar As Integer             'Parada mais alta ou mais baixa antes de atender a chamada
Dim N(1 To 3) As Integer
Dim Pos_Orig(1 To 3) As Integer
Dim Mem(1 To 100, 1 To 2, 1 To 3) As Integer
Dim Custo(1 To 3, 1 To 100) As Integer      'Custo de cada parada agendada
Dim Trag(1 To 3, 1 To 100) As Integer       'Tragetória do elevador
Dim C_EST(1 To 3, 1 To 3) As Integer
Dim J As Integer
Dim J2 As Integer
Dim WT(1 To 3) As Integer
Dim Encontrou As Integer
Dim c As Integer
Dim Pos_Orig_Menor As Integer
Dim Pos_Dest_Menor As Integer
Dim Menor As Integer
Dim Elev As Integer
Dim K As Integer
Dim K2 As Integer
```

```
Dim K2 As Integer
Dim Consecutivo As Integer
Dim Menor_Dest As Integer
Dim Custo_Orig_Existente As Integer
Dim Custo_Dest_Existente As Integer
Dim Custo_Dest(1 To 3) As Integer
Dim Viagem(1 To 3) As Integer
Dim Pos_Dest(1 To 3) As Integer

Tp = 2
Ta = 3
Tac = 3
Tes = 5


'Coletando informações do sistema quando uma nova chamada chega no otimizador
'Número de paradas
N(1) = XL.Workbooks(1).Worksheets(1).cells(1, 1)
N(2) = XL.Workbooks(1).Worksheets(1).cells(2, 1)
N(3) = XL.Workbooks(1).Worksheets(1).cells(3, 1)


'Tragetória e Custo do Elevador 1
For i = 1 To N(1)
    Trag(1, i) = XL.Workbooks(1).Worksheets(1).cells(1, i + 4)
    Custo(1, i) = XL.Workbooks(1).Worksheets(1).cells(2, i + 4)
Next i

'Tragetória e Custo do Elevador 2
For i = 1 To N(2)
    Trag(2, i) = XL.Workbooks(1).Worksheets(1).cells(4, i + 4)
    Custo(2, i) = XL.Workbooks(1).Worksheets(1).cells(5, i + 4)
Next i

'Tragetória e Custo do Elevador 3
For i = 1 To N(3)
    Trag(3, i) = XL.Workbooks(1).Worksheets(1).cells(7, i + 4)
```

```
For i = 1 To N(3)
    Trag(3, i) = XL.Workbooks(1).Worksheets(1).cells(7, i + 4)
    Custo(3, i) = XL.Workbooks(1).Worksheets(1).cells(8, i + 4)
Next i


'msgbox "N(1): " & N(1) & "   N(2): " & N(2) & "   N(3): " & N(3)


Orig = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("A_origem"))
Dest = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("A_destino"))

'Atribuindo valores para a matriz estado
'Direção
C_EST(1, 1) = XL.Workbooks(1).Worksheets(1).cells(6, 1)
C_EST(2, 1) = XL.Workbooks(1).Worksheets(1).cells(7, 1)
C_EST(3, 1) = XL.Workbooks(1).Worksheets(1).cells(8, 1)

'Carga
C_EST(1, 2) = XL.Workbooks(1).Worksheets(1).cells(6, 2)
C_EST(2, 2) = XL.Workbooks(1).Worksheets(1).cells(7, 2)
C_EST(3, 2) = XL.Workbooks(1).Worksheets(1).cells(8, 2)

'Andar Atual
C_EST(1, 3) = XL.Workbooks(1).Worksheets(1).cells(6, 3)
C_EST(2, 3) = XL.Workbooks(1).Worksheets(1).cells(7, 3)
C_EST(3, 3) = XL.Workbooks(1).Worksheets(1).cells(8, 3)


'Descobrindo a quantidade de paradas que cada elevador tem agendado

'msgbox Orig & " => " & Dest

For i = 1 To 3
    Encontrou = 0                   'Resetando a flag
    If Orig < Dest Then             'Chamada ascendente
        If C_EST(i, 1) = 0 Then             'Se o elevador está parado
```

```
        Pos_Orig(i) = 1                 'Se o elevador estiver parado é porque ele já atendeu a todas as chamadas
        If Orig = C_EST(i, 3) Then          'Elevador parado no mesmo andar da origem
            WT(i) = Tp
        Else
            If Orig > C_EST(i, 3) Then      'Elevador parado abaixo da origem
                WT(i) = (Orig - C_EST(i, 3)) * Ta + 2 * Tac + Tp
            Else
                WT(i) = (C_EST(i, 3) - Orig) * Ta + 2 * Tac + Tp
            End If
        End If
    Else
        If C_EST(i, 1) = 1 Then                 'Se o elevador está subindo
            If C_EST(i, 3) < Orig Then          'Elevador subindo, mas abaixo da origem (dá tempo)
                For J = 1 To N(i)               'Verifica se há chamadas ascendentes abaixo da origem
                    If Trag(i, J) < Trag(i, J + 1) Then
                        If Trag(i, J + 1) >= Dest Then
                            If Trag(i, J) = Orig Then
                                WT(i) = Custo(i, J)
                                Encontrou = 1
                                Pos_Orig(i) = J
                                J = N(i) + 1
                            Else
                                If Trag(i, J) < Orig Then
                                    WT(i) = Custo(i, J) + Abs((Orig - Trag(i, J))) * Ta + 2 * Tac + 2 * Tp + Tes
                                    Encontrou = 1
                                    Pos_Orig(i) = J + 1   'Será alocada como a parada seguind à parada j
                                    J = N(i) + 1
                                Else
                                    WT(i) = Custo(i, J) - (Abs((Orig - Trag(i, J))) * Ta + 2 * Tac + 2 * Tp + Tes)
                                    Encontrou = 1
                                    Pos_Orig(i) = J    'Será alocada como a parada seguind à parada j
                                    J = N(i) + 1
                                End If
                            End If
                        Else
                            If N(i) < J + 2 Or Trag(i, J + 2) >= Dest Then
                                If Trag(i, J) > Orig Then
```

```
                                    If Trag(i, J) > Orig Then
                                        WT(i) = Custo(i, J) - (Abs((Orig - Trag(i, J))) * Ta + 2 * Tac + 2 * Tp + Tes)
                                        Encontrou = 1
                                        Pos_Orig(i) = J    'Será alocada como a parada seguind à parada j
                                        J = N(i) + 1
                                    Else
                                        WT(i) = Custo(i, J) + Abs((Orig - Trag(i, J))) * Ta + 2 * Tac + 2 * Tp + Tes
                                        Encontrou = 1
                                        Pos_Orig(i) = J + 1  'Será alocada como a parada seguind à parada j
                                        J = N(i) + 1
                                    End If
                                End If
                            End If
                            If Trag(i, J) > Orig Then   'Verifica até chegar no andar de origem, depois disso não é mais necessário
                                J = N(i) + 1
                            End If
                        End If
                    Next J
                    If Encontrou <> 1 Then
                        WT(i) = Custo(i, N(i)) + Abs((Trag(i, N(i)) - Orig)) * Ta + 2 * Tac + 2 * Tp + Tes
                        Pos_Orig(i) = N(i) + 1      'Simplesmente aloca no final
                    End If
                    If Trag(i, 1) >= Orig And Trag(i, 2) > Trag(i, 1) Or Trag(i, 1) > Trag(i, 2) And Trag(i, 1) >= Dest Then 'Caso esteja
                        WT(i) = (Orig - C_EST(i, 3)) * Ta + Tac + Tp       'Ou então se o elevador estiver subindo para atender a uma cha
                        Pos_Orig(i) = 1         'Novamente a chamada toma a frente das que já estão na memória
                    End If
                Else
                    If C_EST(i, 3) >= Orig Then      'Já passou da origem
                        For J = 1 To N(i)            'Vai vasculhar as chamadas alocadas em cada elevador para tentar encaixar a chamada
                            If Trag(i, J) < Trag(i, J + 1) Then    'Procura se em algum momento o elevador terá uma tragetória ascendente
                                If Trag(i, J) < Orig Then   'Se esta parada está abaixo da origem da chamada
                                    If Trag(i, J + 1) >= Orig Then
                                        WT(i) = Custo(i, J) + Tes + 2 * Tp + 2 * Tac + (Orig - Trag(i, J)) * Ta
                                        Encontrou = 1
                                        Pos_Orig(i) = J + 1
                                        J = N(i) + 1
                                        End If
```

```
                                End If
                            Else
                                If Trag(i, J) = Orig Then
                                    WT(i) = Custo(i, J)
                                    Encontrou = 1
                                    Pos_Orig(i) = J        'O elevador já iria parar naquele lugar, então tem que ser em j para que
                                    J = N(i) + 1
                                End If
                            End If
                        End If
                    Next J
                    If Encontrou <> 1 Then          'Se não conseguiu encaixar a nova chamada
                        WT(i) = Custo(i, N(i)) + Tes + 2 * Tac + 2 * Tp + Abs((Orig - Trag(i, N(i)))) * Ta 'Caso não encontre nenhuma
                        Pos_Orig(i) = N(i) + 1
                    End If
                End If
            End If
        End If
        If C_EST(i, 1) = 2 Then         'Se o elevador está baixando
            If Orig = C_EST(i, 3) And Trag(i, 1) = Orig Then   'Se a próxima parada é no mesmo andar da origem e ele está atualmente
                If Trag(i, 2) > Orig Then               'Se a próxima parada é para cima, se ele vai subir
                    WT(i) = Tp
                    Pos_Orig(i) = 1
                End If
            Else
                If Orig > C_EST(i, 3) Or Orig < C_EST(i, 3) Then    'Se o elevador estiver acima ou abaixo, não faz diferença, o mod
                    For J = 1 To N(i)
                        If Trag(i, J) < Trag(i, J + 1) Then
                            If Trag(i, J) < Orig Then
                                If Trag(i, J + 1) >= Orig Then      'Necessário que o próximo ponto seja acima da origem, caso contr
                                    WT(i) = Custo(i, J) + Tes + 2 * Tp + 2 * Tac + Abs((Orig - Trag(i, J))) * Ta
                                    Encontrou = 1
                                    Pos_Orig(i) = J + 1
                                    J = N(i) + 1
                                End If
                            Else
                                If Trag(i, J) = Orig Then
```

```
                                If Trag(i, J) = Orig Then
                                    WT(i) = Custo(i, J)
                                    Pos_Orig(i) = J
                                    Encontrou = 1
                                    J = N(i) + 1
                                End If
                            Else
                                If Trag(i, J) = Orig Then
                                    WT(i) = Custo(i, J)
                                    Encontrou = 1
                                    Pos_Orig(i) = J
                                    J = N(i) + 1
                                End If
                            End If
                        End If
                    Next J
                    If Encontrou <> 1 Then          'Se não conseguiu encaixar a nova chamada
                        WT(i) = Custo(i, N(i)) + Tes + 2 * Tac + 2 * Tp + Abs((Orig - Trag(i, N(i)))) * Ta 'Caso não encontre nenhuma
                        Pos_Orig(i) = N(i) + 1
                    End If
                End If
            End If
        End If
    End If
Else                            'Chamada descendente
    If C_EST(i, 1) = 0 Then             'Se o elevador está parado
            If Orig = C_EST(i, 3) Then          'Elevador parado no mesmo andar da origem
                WT(i) = Tp
                Pos_Orig(i) = 1
            Else
                WT(i) = Abs((Orig - C_EST(i, 3))) * Ta + 2 * Tac + Tp
                Pos_Orig(i) = 1
            End If
    Else
        If C_EST(i, 1) = 1 Then
            If Orig = C_EST(i, 3) And Trag(i, 1) = Orig Then   'Se a próxima parada é no mesmo andar da origem e ele está atualmente nes
                If Trag(i, 2) < Orig Then               'Se a próxima parada é para baixo, se ele vai descer
                    WT(i) = Tp
                    Pos_Orig(i) = 2             'Primeiro vai atender a chamada no topo, depois vai descer e então passar na origem
                End If
            Else
                If Orig > C_EST(i, 3) Or Orig < C_EST(i, 3) Or Orig = C_EST(i, 3) Then    'Se o elevador estiver acima ou abaixo, não fa
                    For J = 1 To N(i)
                        If Trag(i, J) > Trag(i, J + 1) Then
```

```
                If Trag(i, J) > Trag(i, J + 1) Then
                    If Trag(i, J) > Orig Then
                        If Trag(i, J + 1) <= Orig Then         '10-8 para um Orig = 4 puladria a parada em 8 e atenderia direto a
                            WT(i) = Custo(i, J) + Tes + 2 * Tp + 2 * Tac + Abs((Orig - Trag(i, J))) * Ta
                            Encontrou = 1
                            Pos_Orig(i) = J + 1
                            J = N(i) + 1                        'Garante que na primeira possibilidade de encaixar o loop irá pa
                        End If
                    Else
                        If Trag(i, J) = Orig Then
                            WT(i) = Custo(i, J)
                            Encontrou = 1
                            Pos_Orig(i) = J
                            J = N(i) + 1
                        End If
                    End If
                End If
            Next J
            If Encontrou <> 1 Then          'Se não conseguiu encaixar a nova chamada
                WT(i) = Custo(i, N(i)) + Tes + 2 * Tac + 2 * Tp + Abs((Orig - Trag(i, N(i)))) * Ta 'Caso não encontre nenhuma cha
                Pos_Orig(i) = N(i) + 1
            End If
        End If
    End If
Else
    If C_EST(i, 1) = 2 Then                 'Se o elevador está descendo
            If C_EST(i, 3) > Orig Then         'Elevador descendo, mas acima da origem (dá tempo)
                For J = 1 To N(i)
                    If Trag(i, J) < Trag(i, J + 1) Then     'O elevador vai subir depois
                        If Trag(i, J) <= Dest Then           'Da para encaixar a chamada nesta descida
                            If Trag(i, J) < Orig Then       'Se a origem estiver antes da proxima parada, a origem vai tomar o s
                                WT(i) = Custo(i, J) - (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Descontando
                                Encontrou = 1
                                Pos_Orig(i) = J
                                J = N(i) + 1
                            Else
                                WT(i) = Custo(i, J) + (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Somando o cu
```

```
                                WT(i) = Custo(i, J) + (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Somando o cu
                                Encontrou = 1
                                Pos_Orig(i) = J + 1
                                J = N(i) + 1
                            End If
                        End If
                        If Trag(i, J) > Orig Then
                            J = N(i) + 1
                        End If
                    Else
                        If Trag(i, J + 1) <= Dest Then
                            If Trag(i, J) = Orig Then
                                WT(i) = Custo(i, J)
                                Encontrou = 1
                                Pos_Orig(i) = J
                                J = N(i) + 1
                            Else
                                If Trag(i, J) < Orig Then       'Se a origem estiver antes da proxima parada, a origem vai tomar
                                    WT(i) = Custo(i, J) - (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Descontar
                                    Encontrou = 1
                                    Pos_Orig(i) = J
                                    J = N(i) + 1
                                Else
                                    WT(i) = Custo(i, J) + (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Somando
                                    Encontrou = 1
                                    Pos_Orig(i) = J + 1
                                    J = N(i) + 1
                                End If
                            End If
                        Else
                            If N(i) < J + 2 Or Trag(i, J + 2) <= Dest Then    'Problema com chamada 9-8-3-2 com nova 7-2 que fic
                               If Trag(i, J) < Orig Then       'Se a origem estiver antes da proxima parada, a origem vai tomar
                                    WT(i) = Custo(i, J) - (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Descontar
                                    Encontrou = 1
                                    Pos_Orig(i) = J
                                    J = N(i) + 1
                                Else
```

```
                                    Else
                                        WT(i) = Custo(i, J) + (Abs(Orig - Trag(i, J)) * Ta + 2 * Tac + 2 * Tp + Tes)       'Somando
                                        Encontrou = 1
                                        Pos_Orig(i) = J + 1
                                        J = N(i) + 1
                                    End If
                                End If
                            End If
                        End If
                    Next
                    If Encontrou <> 1 Then
                        WT(i) = Custo(i, N(i)) + Abs((Trag(i, N(i)) - Orig)) * Ta + 2 * Tac + 2 * Tp + Tes
                        Pos_Orig(i) = N(i) + 1
                    End If
                    If Trag(i, J) <= Orig And Trag(i, 2) < Trag(i, 1) And Encontrou <> 1 Or Trag(i, 1) < Trag(i, 2) And Trag(i, 1) =
                        WT(i) = Abs((Orig - C_EST(i, 3))) * Ta + Tac + Tp
                        Pos_Orig(i) = 1
                    End If
                Else
                    If C_EST(i, 3) <= Orig Then      'Já passou da origem
                        For J = 1 To N(i)            'Vai vasculhar as chamadas alocadas em cada elevador para tentar encaixar a char
                            If Trag(i, J) > Trag(i, J + 1) Then    'Procura se em algum momento o elevador terá uma tragetória ascen
                                If Trag(i, J) > Orig Then   'Se esta parada está abaixo da origem da chamada
                                    WT(i) = Custo(i, J) + Tes + 2 * Tp + 2 * Tac + Abs((Orig - Trag(i, J))) * Ta
                                    Encontrou = 1
                                    Pos_Orig(i) = J + 1
                                    J = N(i) + 1
                                Else
                                    If Trag(i, J) = Orig Then
                                        WT(i) = Custo(i, J)
                                        Encontrou = 1
                                        Pos_Orig(i) = J
                                        J = N(i) + 1
                                    End If
                                End If
                            End If
                        Next J
```

```
                        Next J
                        If Encontrou <> 1 Then          'Se não conseguiu encaixar a nova chamada
                            WT(i) = Custo(i, N(i)) + Tes + 2 * Tac + 2 * Tp + Abs((Orig - Trag(i, N(i)))) * Ta 'Caso não encontre ne
                            Pos_Orig(i) = N(i) + 1
                        End If
                    End If
                End If
            End If
        End If
    End If
Next i


'Calculando o tempo de viagem de cada elevador

'Encontrando a posição onde ficaria o destino em cada um dos elevadores
For J = 1 To 3
    For i = Pos_Orig(J) To N(J)
        If Orig < Dest Then         'Subindo
            If Trag(J, i) >= Dest Or Trag(J, i) > Orig And Trag(J, i + 1) < Trag(J, i) Then 'Se o elevador vai subir, mas chega em uma andar
                Pos_Dest(J) = i
                If i = N(J) And Dest > Trag(J, i) Then
                    Pos_Dest(J) = N(J) + 1
                End If
                i = N(J) + 1     'Parar na primeira vez que encontra
            End If
        Else
            If Trag(J, i) <= Dest Or Trag(J, i) < Orig And Trag(J, i + 1) > Trag(J, i) Then 'Se o elevador vai descer, mas chega em um andar
                Pos_Dest(J) = i
                i = N(J) + 1
            End If
        End If
    Next i
Next J

'Problema com chamada agendada 10-8 e com chamada nova de 9-10 onde a posição dos dois ficavam iguais a 1
```

```
'Problema com chamada agendada 10-8 e com chamada nova de 9-10 onde a posição dos dois ficavam iguais a 1
For J = 1 To 3
    If Pos_Orig(J) = Pos_Dest(J) Or Pos_Dest(J) = 0 Then       'Segunda condição porque 9-2 na memória e 9-10 como chamada nova Pos_Dest_Men
        Pos_Dest(J) = Pos_Orig(J) + 1
    End If
Next J

'msgbox "Alocação dos destinos em cada elevador" & vbNewLine & "Pos_Orig(1): " & Pos_Orig(1) & "    Pos_Dest(1): " & Pos_Dest(1) & vbNewLine

'Calculando o custo de cada destino no seu respectivo elevador
For J = 1 To 3
    Custo_Dest(J) = WT(J) + (Pos_Dest(J) - Pos_Orig(J) - 1) * (Tes + 2 * Tp + 2 * Tac)   'Somando os tempos de todas as paradas entre a orig
    Custo_Dest(J) = Custo_Dest(J) + Abs((Dest - Orig)) * Ta + Tes + 2 * Tp + 2 * Tac             'Somando os tempos de viagem apenas dos an
Next J

'msgbox "Custo Destino 1: " & Custo_Dest(1) & "      Custo Destino 2: " & Custo_Dest(2) & "      Custo Destino 3: " & Custo_Dest(3)

'Calculando o tempo de viagem

For J = 1 To 3
    Viagem(J) = Custo_Dest(J) - WT(J)
Next J

'msgbox "WT(1): " & WT(1) & vbNewLine & "WT(2): " & WT(2) & vbNewLine & "WT(3): " & WT(3)
'msgbox "Viagem(1): " & Viagem(1) & vbNewLine & "Viagem(2): " & Viagem(2) & vbNewLine & "Viagem(3): " & Viagem(3)

'Verificando a quantidade de passageiros no elevador, se estiver no limite, é adicionado um custo exorbitante para garantir que o elevador n

For i = 1 To 3
    If C_EST(i, 2) = 8 Then
        WT(i) = WT(i) + 1000
    End If
Next i

'''''''''''''''''''''''''''''''''''''''''''''''
'''''''''''''''''FUNÇÃO CUSTO'''''''''''''''''
'''''''''''''''''''''''''''''''''''''''''''''''
```

```
'''''''''''''''''FUNÇÃO CUSTO'''''''''''''''''
'''''''''''''''''''''''''''''''''''''''''''''''
'Somando ao tempo de espera, o tempo de viagem
For J = 1 To 3
    WT(J) = 0.1 * WT(J) + 0.9 * Viagem(J)
Next J


'Encontrando o menor tempo de espera
If WT(1) < WT(2) Then
    If WT(1) < WT(3) Then
        Menor = WT(1)
        Pos_Orig_Menor = Pos_Orig(1)
        Elev = 1
    Else
        Menor = WT(3)
        Pos_Orig_Menor = Pos_Orig(3)
        Elev = 3
    End If
Else
    If WT(2) < WT(3) Then
        Menor = WT(2)
        Pos_Orig_Menor = Pos_Orig(2)
        Elev = 2
    Else
        Menor = WT(3)
        Pos_Orig_Menor = Pos_Orig(3)
        Elev = 3
    End If
End If

'Caso todos os custos sejam iguais, procura o que tem menos paradas agendadas

'Três custos iguais
If WT(1) = WT(2) And WT(2) = WT(3) Then
    If N(1) < N(2) Then
        If N(1) < N(3) Then
```

```
        If N(1) < N(3) Then
            Menor = WT(1)
            Pos_Orig_Menor = Pos_Orig(1)
            Elev = 1
        Else
            Menor = WT(3)
            Pos_Orig_Menor = Pos_Orig(3)
            Elev = 3
        End If
    Else
        If N(2) < N(3) Then
            Menor = WT(2)
            Pos_Orig_Menor = Pos_Orig(2)
            Elev = 2
        Else
            Menor = WT(3)
            Pos_Orig_Menor = Pos_Orig(3)
            Elev = 3
        End If
    End If
End If

'Dois custos iguais, comparando os outros casos dois a dois
If WT(1) = WT(2) And WT(1) < WT(3) Then
    If N(1) < N(2) Then
        Menor = WT(1)
        Pos_Orig_Menor = Pos_Orig(1)
        Elev = 1
    Else
        Menor = WT(2)
        Pos_Orig_Menor = Pos_Orig(2)
        Elev = 2
    End If
End If

If WT(1) = WT(3) And WT(1) < WT(2) Then
    If N(1) < N(3) Then
```

```
    If N(1) < N(3) Then
        Menor = WT(1)
        Pos_Orig_Menor = Pos_Orig(1)
        Elev = 1
    Else
        Menor = WT(3)
        Pos_Orig_Menor = Pos_Orig(3)
        Elev = 3
    End If
End If

If WT(3) = WT(2) And WT(3) < WT(1) Then
    If N(3) < N(2) Then
        Menor = WT(3)
        Pos_Orig_Menor = Pos_Orig(3)
        Elev = 3
    Else
        Menor = WT(2)
        Pos_Orig_Menor = Pos_Orig(2)
        Elev = 2
    End If
End If

Menor = Menor - Viagem(Elev) 'Resetando o valor de Menor porque antes ele estava somado ao tempo de viagem e isso faria contar o tempo da tr

'Primeira chamada daquele elevador, estava dando problema com os vetores já que o N era zero
If N(Elev) = 0 Then
    'msgbox "Primeira Chamada" & vbNewLine & "Origem: " & Orig & "  Dest: " & Dest & "  Elev: " & Elev
    Trag(Elev, 1) = Orig
    Trag(Elev, 2) = Dest
    Custo(Elev, 1) = Menor
    Custo(Elev, 2) = Menor + Abs(Orig - Dest) * Ta + Tes + 2 * Tac + 2 * Tp
    Pos_Orig_Menor = 1
    Pos_Dest_Menor = 2
    N(Elev) = 2
    GoTo Primeiro
End If
```

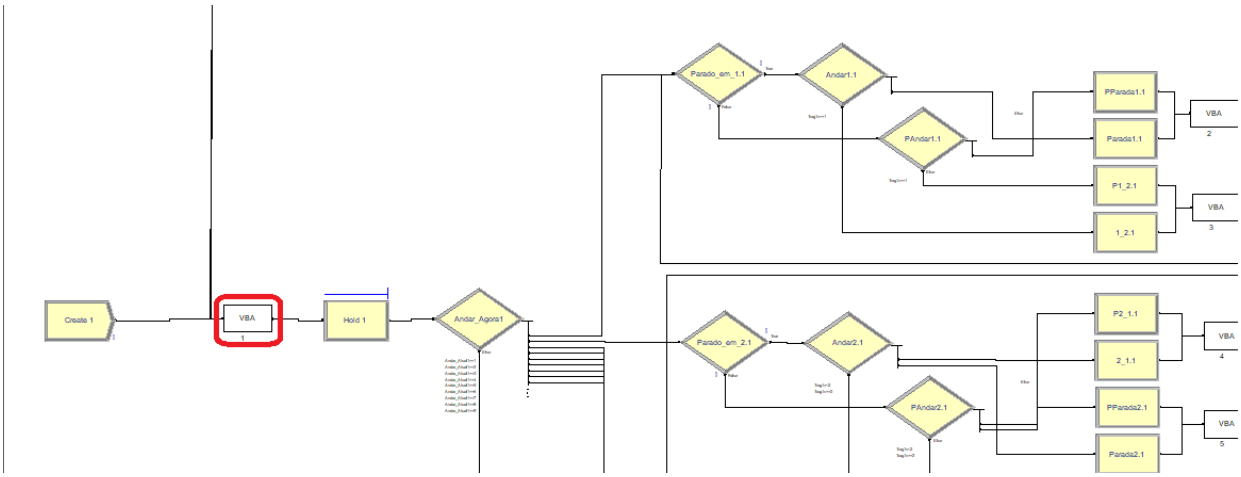
```
End If


'Encontrando a posição do destino no em Trag
For i = Pos_Orig_Menor To N(Elev)
    If Orig < Dest Then          'Subindo
        If Trag(Elev, i) >= Dest Or Trag(Elev, i) > Orig And Trag(Elev, i + 1) < Trag(Elev, i) Then 'Se o elevador vai subir, mas chega em u
            Pos_Dest_Menor = i
            If i = N(Elev) And Dest > Trag(Elev, i) Then                    'Problema devido a chamada na memória 1-3 e a nova 1-5 sendo que
               Pos_Dest_Menor = N(Elev) + 1
            End If
            i = N(Elev) + 1     'Parar na primeira vez que encontra
        End If
    Else
        If Trag(Elev, i) <= Dest Or Trag(Elev, i) < Orig And Trag(Elev, i + 1) > Trag(Elev, i) Then 'Se o elevador vai descer, mas chega em
            Pos_Dest_Menor = i
            i = N(Elev) + 1
        End If
    End If
Next i

'msgbox "Posicao Depois: " & Pos_Dest_Menor

Consecutivo = 0

'Problema com chamada agendada 10-8 e com chamada nova de 9-10 onde a posição dos dois ficavam iguais a 1
If Pos_Orig_Menor = Pos_Dest_Menor Or Pos_Dest_Menor = 0 Then       'Segunda condição porque 9-2 na memória e 9-10 como chamada nova Pos_Des
    Pos_Dest_Menor = Pos_Orig_Menor + 1
End If


'Atualizando os vetores de custo e de tragetória
J = N(Elev)
J2 = N(Elev)
If N(Elev) >= Pos_Orig_Menor Then
```

```
If N(Elev) >= Pos_Orig_Menor Then

        If Trag(Elev, Pos_Orig_Menor) <> Orig Then          'Se não há parada agendada previamente para o andar de origem
            For c = Pos_Orig_Menor To J
                Trag(Elev, J + 1) = Trag(Elev, J)
                J = J - 1
            Next c

            Trag(Elev, J + 1) = Orig
            N(Elev) = N(Elev) + 1
            If Pos_Dest_Menor <> 2 And Dest >= Trag(Elev, Pos_Dest_Menor) And Orig < Dest Or Pos_Dest_Menor <> 2 And Dest <= Trag(Elev, Pos_
                If Trag(Elev, Pos_Dest_Menor) <> Dest Then          'Problema com a chamada 7-2 com Trag = 10-8-2-4-7 que resultava em 10-8-
                    Pos_Dest_Menor = Pos_Dest_Menor + 1
                End If
            End If
        End If
Else        'Caso a posição da origem seja maior que o número de paradas alocadas
    Trag(Elev, Pos_Orig_Menor) = Orig
    N(Elev) = N(Elev) + 1
End If

K = N(Elev)
K2 = N(Elev)
If N(Elev) >= Pos_Orig_Menor Then
    If Trag(Elev, Pos_Dest_Menor) <> Dest Then              'Se não há parada agendada previamente para o andar do destino


        For c = Pos_Dest_Menor To K

                Trag(Elev, K + 1) = Trag(Elev, K)
                'Custo(Elev, K + 1) = Custo(Elev, K)
                K = K - 1
        Next c
        Trag(Elev, K + 1) = Dest
        'Custo(Elev, K + 1) = Menor
        N(Elev) = N(Elev) + 1
    End If
```

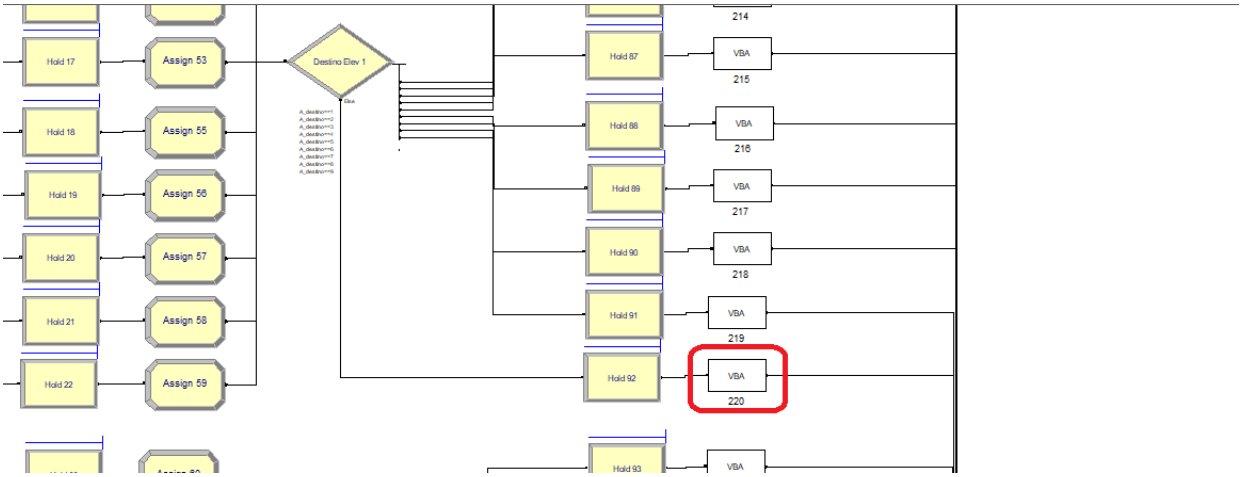
```
    End If
Else
    Trag(Elev, Pos_Dest_Menor) = Dest
    N(Elev) = N(Elev) + 1
End If
'msgbox Orig & " => " & Dest & vbNewLine & "Custo 1: " & WT(1) & vbNewLine & "Custo 2: " & WT(2) & vbNewLine & "Custo 3: " & WT(3) & vbNewLi


'Calculando o custo de chegar até o destino da chamada recém alocada

Menor_Dest = Menor + (Pos_Dest_Menor - Pos_Orig_Menor - 1) * (Tes + 2 * Tp + 2 * Tac)   'Somando os tempos de todas as paradas entre a origei
''msgbox "Custo apenas com as paradas entre: " & Menor_Dest
Menor_Dest = Menor_Dest + Abs((Dest - Orig)) * Ta + Tes + 2 * Tp + 2 * Tac              'Somando os tempos de viagem apenas dos andares da o

'msgbox "Custo até o destino: " & Menor_Dest

'Alocando os novos custos no vetor de custos

Custo_Orig_Existente = 0
Custo_Dest_Existente = 0

K2 = N(Elev)

'Verifica se o custo da origem ou do destino já estão no vetor na posição que deveriam estar, se estiver, se já estiverem, não é necessário

If Custo(Elev, Pos_Orig_Menor) = Menor Then
    Custo_Orig_Existente = 1
End If
If Custo(Elev, Pos_Dest_Menor) = Menor_Dest Then
    Custo_Dest_Existente = 1
End If


'Inserindo o custo da origem no vetor
    If Custo(Elev, K2) = 0 And Custo(Elev, K2 - 1) <> 0 And Custo_Orig_Existente = 0 Then             'Se apenas o último elemento é igual
        For c = Pos_Orig_Menor To K2
            Custo(Elev, K2 + 1) = Custo(Elev, K2)
```

```
        For c = Pos_Orig_Menor To K2
            Custo(Elev, K2 + 1) = Custo(Elev, K2)
            K2 = K2 - 1
        Next c
    Custo(Elev, K2 + 1) = Menor
    End If                                                          'A princípio não é necessária essa terceira condição porque se serão ins

K2 = N(Elev)                    'Resetando a variável

If K2 > 2 Then                  'Garantindo que não vai tentar acessar o endereço 0 do vetor
    If Custo(Elev, K2) = 0 And Custo(Elev, K2 - 1) = 0 And Custo(Elev, K2 - 2) <> 0 And Custo_Orig_Existente = 0 Then 'Se os dois últimos el
        K2 = K2 - 1                                                 'Apenas o primeiro valor será escrito nesse momento, então não pode ir até
        For c = Pos_Orig_Menor To K2
            Custo(Elev, K2 + 1) = Custo(Elev, K2)
            K2 = K2 - 1
        Next c
    Custo(Elev, K2 + 1) = Menor
    End If
End If

K2 = N(Elev)

'Inserindo o custo do destino no vetor
If Custo(Elev, K2) = 0 And Custo(Elev, K2 - 1) <> 0 And Custo_Dest_Existente = 0 Then 'teoricamente não precisa desta última condição porque
    For c = Pos_Dest_Menor To K2
        Custo(Elev, K2 + 1) = Custo(Elev, K2)
        K2 = K2 - 1
    Next c                                  'Não é necessário verificar se há duas vagas porque se tinham, uma já foi preenchida pelo cu
    Custo(Elev, K2 + 1) = Menor_Dest
    Custo_Dest_Existente = 1                'Só para garantir que não vai entrar no próximo if caso entre nesse
End If


'Pode ser que o vetor já esteja completo, mas ainda seja necessário atualizar um valor
If Custo(Elev, K2) <> 0 And Custo_Dest_Existente = 0 Then
    Custo(Elev, Pos_Dest_Menor) = Menor_Dest
```

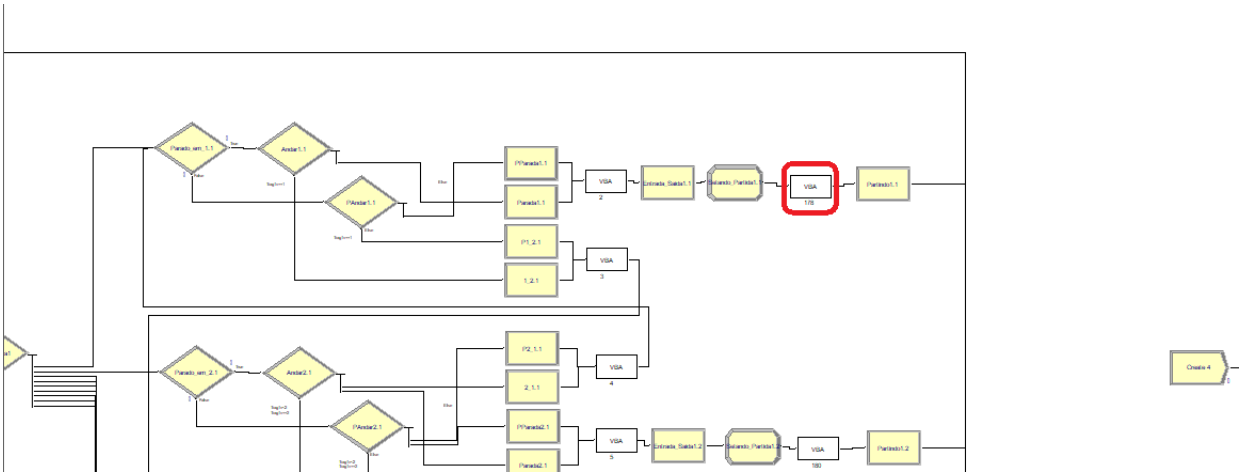
```
    Custo(Elev, Pos_Dest_Menor) = Menor_Dest
End If
Primeiro:

'msgbox "Tragetória"
For i = 1 To N(Elev)
    'msgbox Trag(Elev, i)
Next i

'msgbox "Custo"
For i = 1 To N(Elev)
    'msgbox Custo(Elev, i)
Next i

'Atualizando os dados
XL.Workbooks(1).Worksheets(1).cells(Elev, 1) = N(Elev)      'Número de paradas


If Elev = 1 Then
    For i = 1 To N(1)
        XL.Workbooks(1).Worksheets(1).cells(1, i + 4) = Trag(1, i)      'Tragetória
        XL.Workbooks(1).Worksheets(1).cells(2, i + 4) = Custo(1, i)     'Custo
        s.VariableArrayValue(s.SymbolNumber("Trag1v")) = Trag(1, 1) 'Atualizando a entidade para poder liberar o hold
    Next i
End If

If Elev = 2 Then
    For i = 1 To N(2)
        XL.Workbooks(1).Worksheets(1).cells(4, i + 4) = Trag(2, i)      'Tragetória
        XL.Workbooks(1).Worksheets(1).cells(5, i + 4) = Custo(2, i)     'Custo
        s.VariableArrayValue(s.SymbolNumber("Trag2v")) = Trag(2, 1) 'Atualizando a entidade para poder liberar o hold
    Next i
End If

If Elev = 3 Then
    For i = 1 To N(3)
        XL.Workbooks(1).Worksheets(1).cells(7, i + 4) = Trag(3, i)      'Tragetória
```




```
Private Sub VBA_Block_1_Fire()
Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN
Dim Andar As Integer
Dim Custo As Integer


s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Andar_Atual1")) = XL.Workbooks(1).Worksheets(1).cells(6, 3)    'Atribuindo valor do andar
s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Estado_Atual1")) = XL.Workbooks(1).Worksheets(1).cells(6, 1)    'Atribuindo o estado atual
s.VariableArrayValue(s.SymbolNumber("Trag1v")) = XL.Workbooks(1).Worksheets(1).cells(1, 5)          'Atribuindo valor da próxima parada para
Andar = XL.Workbooks(1).Worksheets(1).cells(1, 5)
Custo = XL.Workbooks(1).Worksheets(1).cells(2, 5)
If Custo < 0 Then                   'Quando tava na volta final (N = 0), o custo dava negativo
    XL.Workbooks(1).Worksheets(1).cells(2, 5) = 0
End If
Custo = XL.Workbooks(1).Worksheets(1).cells(2, 5)
'msgbox "Andar1: " & Andar & vbNewLine & "Custo1: " & Custo

End Sub
```

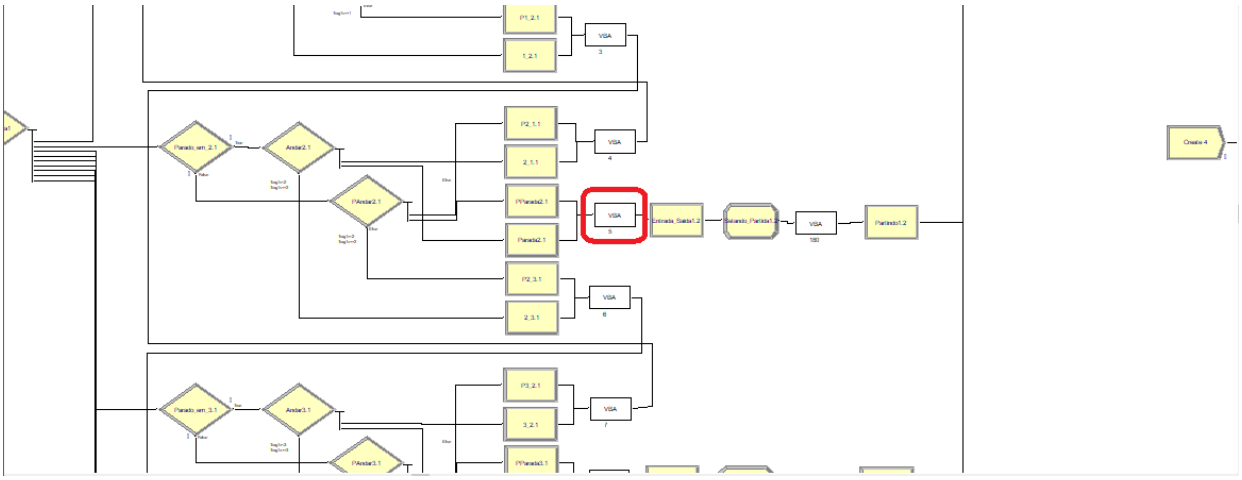



```
Private Sub VBA_Block_220_Fire()
Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN

'Atualizando a carga do elevador
XL.Workbooks(1).Worksheets(1).cells(6, 2) = XL.Workbooks(1).Worksheets(1).cells(6, 2) - 1

End Sub
```

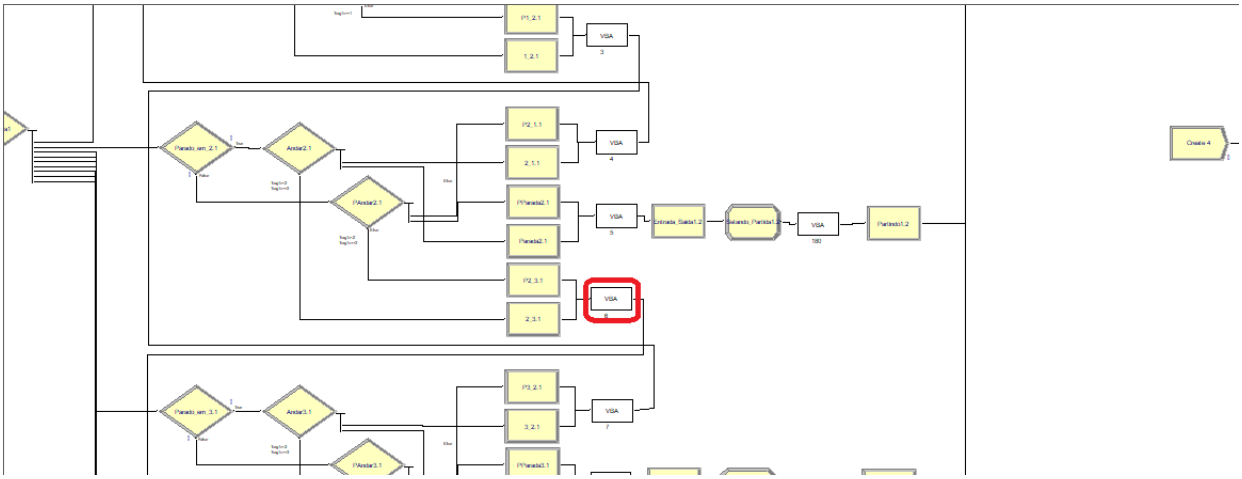



```
Private Sub VBA_Block_178_Fire()
Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN
Dim c As Integer
Dim N As Integer
Dim Custo As Integer


N = XL.Workbooks(1).Worksheets(1).cells(1, 1)       'Lendo o número de paradas agendadas

Custo = XL.Workbooks(1).Worksheets(1).cells(2, 5)   'Lendo o custo que será descontado


'Dando os shifts
For c = 1 To N
    XL.Workbooks(1).Worksheets(1).cells(1, c + 4) = XL.Workbooks(1).Worksheets(1).cells(1, c + 5)         'Deslocando tragetória
    XL.Workbooks(1).Worksheets(1).cells(2, c + 4) = XL.Workbooks(1).Worksheets(1).cells(2, c + 5)         'Deslocamento custo
Next c

s.VariableArrayValue(s.SymbolNumber("Traglv")) = XL.Workbooks(1).Worksheets(1).cells(1, 5)     'Atualizando próxima parada(Trag) depois do sh

XL.Workbooks(1).Worksheets(1).cells(1, 1) = N - 1   'Atualizando número de paradas


'Atualizando a direção do elevador
If XL.Workbooks(1).Worksheets(1).cells(1, 5) > 1 Then
    XL.Workbooks(1).Worksheets(1).cells(6, 1) = 1
Else
    XL.Workbooks(1).Worksheets(1).cells(6, 1) = 2
End If


If N = 0 Then
    s.VariableArrayValue(s.SymbolNumber("Traglv")) = 0
    GoTo final
```

```
Custo = XL.Workbooks(1).Worksheets(1).cells(2, 5)    'Lendo o custo que será descontado


'Dando os shifts
For c = 1 To N
    XL.Workbooks(1).Worksheets(1).cells(1, c + 4) = XL.Workbooks(1).Worksheets(1).cells(1, c + 5)          'Deslocando tragetória
    XL.Workbooks(1).Worksheets(1).cells(2, c + 4) = XL.Workbooks(1).Worksheets(1).cells(2, c + 5)          'Deslocamento custo
Next c

s.VariableArrayValue(s.SymbolNumber("Traglv")) = XL.Workbooks(1).Worksheets(1).cells(1, 5)      'Atualizando próxima parada(Trag) depois do sh

XL.Workbooks(1).Worksheets(1).cells(1, 1) = N - 1    'Atualizando número de paradas



'Atualizando a direção do elevador
If XL.Workbooks(1).Worksheets(1).cells(1, 5) > 1 Then
    XL.Workbooks(1).Worksheets(1).cells(6, 1) = 1
Else
    XL.Workbooks(1).Worksheets(1).cells(6, 1) = 2
End If



If N = 0 Then
    s.VariableArrayValue(s.SymbolNumber("Traglv")) = 0
    GoTo final
End If

For c = 1 To N
    XL.Workbooks(1).Worksheets(1).cells(2, c + 4) = XL.Workbooks(1).Worksheets(1).cells(2, c + 4) - Custo    'Descontando o custo da parada qu
Next c

final:


End Sub
```




```
Private Sub VBA_Block_5_Fire()
Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN
Dim c As Integer
Dim N As Integer
Dim Custo As Integer


'Atualizando valores e fazendo os deslocamentos

s.VariableArrayValue(s.SymbolNumber("Andandol")) = 0    'O elevador está parado e apto para os passageiros subirem ou descerem

XL.Workbooks(1).Worksheets(1).cells(6, 3) = 2       'Atualizando o andar atual

s.VariableArrayValue(s.SymbolNumber("Andar_Atuallv")) = 2    'Setando variável da simulação

End Sub
```




```
Private Sub VBA_Block_6_Fire()
'Atualizando apenas os valores porque o elevador está apenas de passagem por esse andar, não há parada agendada nele
Dim s As SIMAN
Set s = ThisDocument.Model.SIMAN
XL.Workbooks(1).Worksheets(1).cells(6, 1) = 1       'Atualizando direção
XL.Workbooks(1).Worksheets(1).cells(6, 3) = 3       'Atualizando andar atual
s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("Estado_Atual1")) = 1 'Atualizando o atributo direção na simulação
End Sub
```